Modelagem UML no Delphi XE3: Aproveite o melhor para sistemas RAD

Clube Delphi

Gerador de Etiquetas com Fast Report
Recursos avançados que geram flexibilidade

Ano 13 :: Edição 149 :: R$ 11,90

DevMedia

iPhone e Delphi

Desenvolvimento RAD para seu dispositivo

Certificação Delphi
Dicas para se tornar um Delphi Master Developer

Delphi e RIA
Interfaces ricas para WEB, com Delphi

ISSN 1517990-7

DevMedia

DÊ UM SALTO EM CONHECIMENTO!

Acesse o maior portal para desenvolvedores da América Latina!

20 mil posts

430 mil cadastrados

10 milhões de page-views por mês

http://www.devmedia.com.br/

DevMedia

Sumário

Delphi

Novidades

06 – Desenvolvimento para iOS com Delphi XE2
[*Victory Fernandes*]

Solução Completa

16 – Editor de Etiquetas com FastReport
[*Rafael Isique Guardieri Brendolan*]

Boas Práticas

Curso

23 – Certificação Delphi – Parte 2
[*Fabrício Hissao Kawata*]

Curso

31 – RIA com Delphi – Parte 2
[*Giuliano Scombatti Pinto*]

Mentoring

Boas Práticas

44 – Modelagem UML para Projeto não Orientado a Objetos
[*Paulo Quicoli*]

Olá, eu sou o **DevMan**! Desta página em diante, eu estarei lhe ajudando a compreender com ainda mais facilidade o conteúdo desta edição. Será um prazer contar com sua companhia! Confira abaixo o que teremos nesta revista:

Dê seu feedback sobre esta edição!

A ClubeDelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre esta edição, artigo por artigo, através do link:
www.devmedia.com.br/clubedelphi/feedback

RENOVE JÁ!

Sua assinatura pode estar acabando

Renovando a assinatura de sua revista favorita você ganha brindes e descontos exclusivos.

www.devmedia.com.br/renovacao ou (21)3382-5038

DevMedia

Clube Delphi

Ano 13 - 149ª Edição 2013 - ISSN 1517990-7 Impresso no Brasil

Corpo Editorial

Editor Geral

Paulo Quicoli
pauloquicoli@gmail.com

Editor Técnico

Daniel Sobrinho Laporte
daniel.laporte@gmail.com

Equipe Editorial

Guinther Pauli
guintherpauli@gmail.com

Fabrício Hissao Kawata
fabricio.kawata@bol.com.br

Giuliano Scombatti Pinto
giuliano@sygnux.com.br

Jornalista Responsável

Kaline Dolabella - JP24185

Atendimento ao Leitor

A DevMedia conta com um departamento exclusivo para o atendimento ao leitor. Se você tiver algum problema no recebimento do seu exemplar ou precisar de algum esclarecimento sobre assinaturas, exemplares anteriores, endereço de bancas de jornal, entre outros, entre em contato com:

www.devmedia.com.br/central
(21) 3382-5038

Publicidade

Para informações sobre veiculação de anúncio na revista ou no site e para fechar parcerias ou ações específicas de marketing com a DevMedia, entre em contato com:

publicidade@devmedia.com.br

Distribuição

FC Comercial e Distribuidora S.A
Rua Teodoro da Silva, 907
Grajaú - RJ - 206563-900

EDITORIAL

A corrida para o desenvolvimento móvel só está no início. Vemos muitas plataformas se adequando a essa realidade e principalmente, o mercado tem exigido isso. Nós desenvolvedores Delphi não estamos sozinhos nessa. A Embarcadero disponibilizou na versão XE2 do Delphi um mecanismo que permite o desenvolvimento para a plataforma iOS. Nesta edição o Victory nos apresenta uma sugestão de ambiente de desenvolvimento e mostra na prática, como criar seu primeiro aplicativo para iPhone.

E falando ainda de desenvolvimento móvel, o Giuliano traz a última parte de seu mini curso sobre desenvolvimento RIA. Veremos como instalar um aplicativo no Apache e conheceremos outro framework, chamado Raudus. Sem dúvida o desenvolvimento hoje não se restringe apenas ao desktop, assim, desenvolvedores, vamos nos atualizar com novos conceitos, sem deixar o RAD de lado.

E por falar em RAD, o Rafael apresenta uma solução que vai garantir uma flexibilidade enorme para aqueles velhos pedidos de alteração de layout de etiquetas. Como agora o Delphi traz consigo uma versão do FastReport, porque não explorar suas funcionalidade e criar algo dinâmico? É exatamente isso que é apresentado, um editor de etiquetas. Não perca.

O Fabrício apresenta o fim de um assunto muito interessante e que sempre gera discussão. Certificações. Mas agora ele traz o nível mais profundo da certificação Delphi, a Master Developer. Quer ser mestre? Leia esse artigo e veja as dicas.

Por fim, eu trago algo que sempre deixamos passar. Documentação. Mas não qualquer documentação. Apresento a UML para desenvolvedores RAD. A principal dificuldade de adoção da UML estava no fato do diagrama de classes. Como utilizá-lo se meu sistema não é OO? A resposta é simples, não utilize! Veja outros diagramas que darão maior visibilidade e entendimento para as funções existentes em seus projetos, e o mais importante, todos são criados a partir do Delphi XE3.

Boa leitura.

Paulo Quicoli
pauloquicoli@gmail.com
Editor Geral
quicoli.wordpress.com
twitter.com/pauloquicoli

Fale com o Editor!

É muito importante para a equipe saber o que você está achando da revista: que tipo de artigo você gostaria de ler, que artigo você mais gostou e qual artigo você menos gostou. Fique a vontade para entrar em contato com os editores e dar a sua sugestão!

Se você estiver interessado em publicar um artigo na revista ou no site ClubeDelphi, entre em contato com os editores, informando o título e mini-resumo do tema que você gostaria de publicar:

Paulo Quicoli - Editor da Revista
pauloquicoli@gmail.com

MVP

Assine agora e tenha acesso a todo o conteúdo da DevMedia: **www.devmedia.com.br/mvp**

DevMedia

Seu **conhecimento vale muito** na DevMedia

Publique artigos e vídeos gratuitamente no site e participe do programa **PayContent.**

Sua participação pode ser premiada: quanto mais você postar, mais chances de ganhar.

Para saber mais acesse:
www.devmedia.com.br/paycontent

DevMedia

Desenvolvimento para iOS com Delphi XE2

Aplicações para dispositivos móveis nunca foram tão simples

O mercado de aplicações móveis vem crescendo continuamente e o Brasil atualmente é um dos maiores consumidores do mundo. Dentre todo esse "boom" mercadológico, vários desenvolvedores iniciaram projetos e investimento em aplicações para o mundo móvel. Dentre todas as plataformas atualmente existentes, três em particular vem ganhando um maior destaque sendo elas: o Android, iOS e Windows Phone.

Como se sabe, o Android é a plataforma da Google, que domina hoje a maior parte dos dispositivos móveis do mercado, rodando nos mais variados tipos de dispositivos (Tablets e Smartphones) e tem seu desenvolvimento nativo baseado na plataforma Java com o SDK Android (**Nota do DevMan 1**) além de uma extensa gama de IDEs como o NetBeans e o Eclipse por exemplo. Já o iOS é o Sistema Operacional oficial e exclusivo da Apple presente nos SmartPhones iPhone e no Tablet iPad e que adota como linguagem nativa de desenvolvimento o Object-C ou Objective-C além de seu IDE, também exclusivo, o xCode. E por último, o Windows Phone que é propriedade da Microsoft e traz suporte através de várias linguagens suportadas pelo IDE do Visual Studio como C#, VB.Net entre tantas outras, baseando-se no desenvolvimento através do Silverlight.

Nota do DevMan 1

SDK: são as siglas de "Software Development Kit" que como o próprio nome sugere, é um conjunto de tecnologias para desenvolvimento. São frequentemente utilizados em Java (JSDK), Android entre outras das mais comuns tecnologias do mercado.

Desta forma, pensando no futuro, a Embarcadero dedicou-se a suportar, integrar e incorporar estes ambientes por meio de sua principal suíte de desenvolvimento, o RAD Studio. Sendo assim, o Android (e outras plataformas) foram abrangidas pelo desenvolvimento por meio do RADPHP que utiliza-se do SDK Android e PhoneGap/Cordova (**Nota do DevMan 2**) para a criação de aplicações híbridas (**Nota do Devman 3**). Para o desenvolvimento voltado ao Windows Phone a solução que ofereceu suporte nativo foi o PrismXE2, que por meio do compilador Oxygene, oferece a possibilidade de criação de aplicações WP7 (Windows Phone 7) dentro do Visual Studio com Silverlight e por último, o carro chefe da Embarcadero, o Delphi XE2 que veio com uma proposta inovadora na qual permitia pela primeira vez, a criação de aplicações para Mac por meio do FireMonkey (também lançado nesta versão), suportando assim a criação de Apps para os ambientes iOS (iPhone/iPad) e MacOSX.

Embora esta mudança tenha desagradado uma parte da comunidade Delphi, o lançamento do Delphi XE3 marca um ponto

Resumo DevMan

Desenvolvimento para iOS com Delphi XE2:

O mercado de aplicativos móveis está em alta e em constante crescimento. Como se sabe, na versão XE2 o Delphi passou a suportar o desenvolvimento não só para Windows, mas sim para os ambientes MAC e iOS. Tudo isso foi possível graças ao FireMonkey que permitiu a criação de aplicações para estas plataformas de uma forma nativa. Neste artigo, veremos passo a passo como criar sua primeira aplicação para dispositivos móveis de forma nativa, envolvendo etapas de configuração de ambiente, compilação, execução e toda a cobertura de detalhes para que você possa ingressar neste modelo de desenvolvimento.

Em que situação o tema é útil:

O tema é útil para desenvolvedores que estão iniciando no desenvolvimento Mobile para a plataforma iOS ou mesmo para quem deseja criar aplicações para iPad ou MAC. Para isso, são apresentados componentes específicos desenvolvidos pela Embarcadero que auxiliam o desenvolvedor nas tarefas de comunicação com recursos nativos do dispositivo.

chave na estratégia da Embarcadero. Com a finalidade de criar uma ferramenta definitiva para o desenvolvimento móvel, a Embarcadero acabou "desacoplando" a parte de desenvolvimento Mobile que integrava o Delphi XE2 e FireMonkey, embora a criação de aplicações para MacOSX ainda fosse suportada de forma nativa na versão XE3.

Nota do DevMan 2

PhoneGap: PhoneGap ou Cordova como também é conhecido, é um Framework hibrido (ver **Nota do DevMan 3**) que une as tecnologias HTML, CSS e JavaScript, permitindo assim um desenvolvimento multiplataforma. Sua proposta é escrever o código uma única vez e permitir a criação da aplicação nas principais plataformas móveis do mercado. Possui um compilador online (Adobe Phonegap Build – ver seção **Links**) que permite a depuração no próprio emulador de acordo com a plataforma selecionada.

Nota do DevMan 3

Aplicações híbridas : Ao contrário de aplicações nativas que tem como a principal vantagem e característica a utilização total dos recursos disponíveis no dispositivo móvel, as aplicações híbridas são nada mais do que Frameworks que encapsulam diversas tecnologias com a finalidade de permitir desenvolver para múltiplas plataformas móveis com o mesmo código. Geralmente baseiam-se em linguagens como HTML, CSS e JavaScript (jQueryMobile) para realizar o trabalho de efeitos visuais, entre outros. A principal vantagem é a curva de aprendizado menor, já que para quem tem conhecimento de web, pode facilmente desenvolver usando esta abordagem. A grande desvantagem é que nem sempre os recursos dos Frameworks híbridos irão suportar os recursos mais novos do dispositivo, no entanto, é um conceito que vem crescendo muito entre o desenvolvimento móvel.

Toda essa mudança dará origem (em breve), a mais nova suíte de desenvolvimento, o Mobile Studio. Assim como parte desta mudança, o lançamento do XE3 também marca o fim da era RADPHP, dando lugar ao HTML5 Builder, na qual se tornou seu sucessor e que continuará abordando o desenvolvimento Web/Mobile de forma hibrida utilizando-se de linguagens e tecnologias como o HTML5, CSS, JavaScript, jQueryMobile entre tantos outros.

No entanto, se você ainda não possui o Delphi XE3 e deseja criar aplicações para dispositivos móveis da Apple, até o presente momento é possível desenvolver para dispositivos Macintosh/Apple adotando o Delphi XE2 e o FireMonkey, utilizando como linguagem principal o Object Pascal, na qual o leitor já está ambientado, o que permite uma produtividade muito maior, já que o desenvolvimento das telas e a programação das regras é extremamente intuitivo e a curva de aprendizado é fantástica!

Vale ressaltar que além de todos estes fatores, ainda é possível criar aplicações DataSnap/REST (**Nota do DevMan 4**) que permitem uma comunicação simples e rápida entre dispositivos e Web, o que sem dúvida, é algo de extrema usabilidade para quem já possui aplicações e quer desenvolver módulos a parte que suportem as necessidades do mundo móvel.

Sendo assim, neste artigo cobriremos todo o processo de configuração da plataforma incluindo detalhes para criação dos seus primeiros aplicativos com telas, botões, campos de texto e imagens, além de realizar os primeiros testes de simulação e verificação de funcionalidade da solução.

Utilizaremos como base os aplicativos de demonstração disponibilizados por **Anders Ohlsson** da Embarcadero que mostram como desenvolver aplicativos com acesso a recursos básicos, bem como recursos de hardware e que estão disponíveis para download na seção **Links**, ao final deste artigo.

Nota do DevMan 4

Aplicações DataSnap e REST: O framework DataSnap do Delphi permite a criação de uma arquitetura em camadas. Essa arquitetura em camadas é uma das mais utilizadas em sistemas de grande porte, pois permite ter um sistema mais escalável e de fácil manutenção porque centraliza regras de negócio em um servidor chamado "servidor de aplicação". O REST permite que um servidor de aplicação possa ser consumido via web aumentando consideravelmente seu nível de escalabilidade e disponibiliza suas funcionalidades como serviços web de forma simples, independente de tecnologia

O Delphi XE2 será tema do nosso artigo como IDE escolhido para realizar o desenvolvimento do aplicativo para iOS. Sendo assim, vamos precisar de um ambiente Windows configurado, para instalar o Delphi XE2 e compilar nossa aplicação.

Independente da plataforma de programação escolhida é preciso dispor de um ambiente MacOS para recompilar sua aplicação no xCode (**Nota do DevMan 5**) de forma a gerar código compatível e pronto para ser testado em iPhone e/ou iPAD.

Nota do DevMan 5

xCode: O xCode é o ambiente de desenvolvimento oficial da Apple, gratuito e disponível para download na loja da Apple (link na seção **Links**). Somente através dele é possível fazer a compilação final necessária para realizar testes da sua aplicação em dispositivos reais, após concluídos os testes nos emuladores.

Já que precisaremos dos dois ambientes, Windows e MacOS instalados e configurados, o melhor que podemos fazer é instalar em um mesmo computador os dois Sistemas Operacionais para rodar o Delphi XE2 e o xCode respectivamente. Para isso, podemos usar uma solução com instalação em modo DualBoot ou utilizar uma Virtual Machine (**Nota do DevMan 6**) .

Instalando Virtual Machine no Mac e configurando o sistema para gerar aplicações iOS com DelphiXE2

Vamos à preparação do ambiente. Por se tratar de algo que ainda é motivo de dificuldade para grande parte dos usuários, iremos explanar estes passos através da criação de uma máquina virtual.

A instalação em modo DualBoot traz como vantagem a maior estabilidade e performance do computador, uma vez que somente

um Sistema Operacional será carregado por vez. No entanto, a principal desvantagem neste caso é que durante o processo de desenvolvimento pode ser necessário alternar inúmeras vezes entre o Delphi XE2 e o xCode para realizar compilações nos dois SOs e neste caso, esta opção não é nada aconselhada. A instalação de uma VM não traz tanta estabilidade e performance, afinal os dois Sistemas Operacionais estarão carregados ao mesmo tempo, compartilhando por exemplo, a memória RAM da máquina. Em compensação é possível alternar muito facilmente entre as janelas dos aplicativos nos diferentes Sistemas Operacionais, inclusive realizando compartilhamento da pasta do projeto entre o Delphi XE2 e o xCode. Sendo assim, esta opção é a opção mais aconselhada.

Nota do DevMan 6

Virtual Machine: Virtual Machines ou máquinas virtuais, são a grosso modo "emuladores" que permitem utilizar o recurso de virtualização para criar múltiplos ambientes dentro de um Sistema Operacional. Deste modo, é possível rodar diferentes Sistemas Operacionais dentro de uma mesma máquina sem a necessidade de formatação ou hardware específico (como é o caso do MAC por exemplo). A grande desvantagem é o alto consumo de recursos da máquina como disco rígido, memória, processamento e etc. São comumente utilizadas para criar ambientes de testes.

Neste cenário temos as seguintes alternativas para otimizar o trabalho com as duas ferramentas em um único computador utilizando VMs, são elas:

Em uma máquina MAC com xCode 4.2 instalado (**Links**), instalar uma VM com o Windows, instalar o Delphi XE2 e desenvolver seu aplicativo compartilhando o diretório do projeto entre a VM_Windows e o Host_MacOs, possibilitando a compilação do código nas duas plataformas, de forma simples e rápida.

Em uma máquina Windows com Delphi XE2, instalar uma com MacOS, instalar o xCode e desenvolver seu aplicativo compartilhando o diretório do projeto entre a VM_MacOs e o Host_Windows, possibilitando a compilação do código nas duas plataformas.

A escolha do cenário vai depender inicialmente do fato de você dispor ou não de um computador MAC. Caso não disponha, a escolha natural é pelo segundo cenário.

Neste artigo será utilizado o primeiro caso. Veremos como é fácil levantar uma VM no computador, instalar o Windows, o Delphi XE2 e realizar as configurações necessárias para iniciar o desenvolvimento dos seus aplicativos iOS. Vale ressaltar que durante os testes, foi utilizado um MacBook Air com tela de 13", processador Intel Core i5 de 1.7GHz, 4GB de memória RAM e o Sistema Operacional Mac OS X Lion versão 10.7.4 (11E53).

Sem dúvida, a melhor plataforma de VMs no MAC é o VMWare (ver seção **Links**), no entanto, a versão VMWare Fusion 4 compatível com o MAC que é paga e custa US$49,99. O Parallels é outra plataforma bastante adotada por usuários MAC, porém também é uma opção paga que custa US$79,99.

Como alternativa gratuita foi escolhido VirtualBox, adquirido pela ORACLE e disponibilizado sob licença GNU (General Public License - GPL). O VirtualBox roda em hosts Windows, Linux, Mac e Solaris, além de suportar a criação de VMs Windows (diversas versões), DOS, Linux, Solaris, OpenSolaris, OS/2 e OpenBSD. O arquivo adotado foi: "VirtualBox4.1.18-78361-OSX.dmg" disponível no link de downloads do site do VirtualBox.

Ao executar o VirtualBox pela primeira vez surgirá a janela mostrada na **Figura 1**.

Figura 1. Janela principal do VirtualBox após a instalação

Para criar uma nova VM é preciso clicar no botão New e seguir confirmando as opções do Wizard. As configurações sugeridas: VM de Windows XP, pois é bastante estável e consome pouca memória, alocando 1,5GB de RAM e 30 GB de HD para uso, em modo VDI (VirtualBox Disk Image) - Dinamically Allocated. Não se preocupe que todas estas opções serão apresentadas no Wizard de forma bem intuitiva.

Nota

A alocação de menos de 20GB de HD para a VM pode resultar em espaço insuficiente para instalação do Windows+Delphi XE2.

Depois de configurada a VM, será necessário o CD original de instalação do Windows XP ou arquivo .ISO, que será solicitado para proceder com a instalação.

Dando sequência com o processo de instalação, o que temos é uma janela aberta dentro do MAC onde é possível ver a tela de instalação do Windows XP. Basta seguir normalmente os passos e após a conclusão da instalação do Windows e instalar normalmente o Delphi XE2 dentro da VM.

Com a VM instalada podemos compartilhar diretórios entre o host MacOs e a VM Windows. Para compartilhar diretórios o primeiro passo é acessar no menu principal do MacOS a opção Devices> Install Guest Additions e seguir o Wizard de instalação que permite o compartilhamento de diretórios. Após concluir esta

instalação na VM, devemos clicar com o botão direito no pequeno ícone Shared Folder localizado no canto direito inferior da janela da VM conforme mostrado na **Figura 2**.

Figura 2. Configurando a opção de Shared Folders da VM

Será apresentada a janela Shared Folder (**Figura 3**) onde é possível adicionar os diretórios que se deseja compartilhar. Neste exemplo, adicionei um diretório "victory" localizado no desktop do MacOS. Marcando as opções de Auto-Mount e Make Permanent garantimos que o diretório estará acessível sempre que a VM for iniciada.

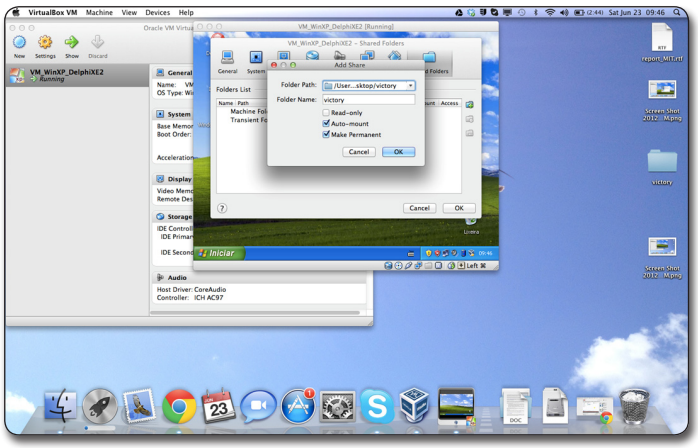

Figura 3. Configurando diretório compartilhado entre a VM e o host

Uma vez compartilhado o diretório no VirtualBox, deve-se acessar o Windows Explorer na VM e no menu principal e escolher a opção Ferramentas>Mapear Unidade de Rede. Na janela de mapeamento de unidade de rede digite \\VboxSvr\NomeDoDiretorioCompartilhado no campo da pasta a ser compartilhada. Neste exemplo, como o nome do diretório compartilhado é victory, foi digitado \\VBoxSvr\victory.

Não podemos esquecer que o objetivo de todo o trabalho feito até aqui é poder criar uma aplicação iOS no Delphi XE2, compilá-la, preparar os arquivos para compilação no xCode e então re-compilar a aplicação no xCode do MacOS, de forma que será então possível testá-la no iPhone e/ou iPAD. Sendo assim, uma vez tendo compartilhado o diretório entre os dois sistemas, devemos abrir o Delphi XE2 que foi instalado na VM e no menu principal do iniciar um projeto novo (File> New> Other e FireMonkey HD iOS Application).

Surgirá então o formulário com o visual do iPhone e nele adicionaremos um componente TEdit e um TButton, conforme mostrado na **Figura 4.**

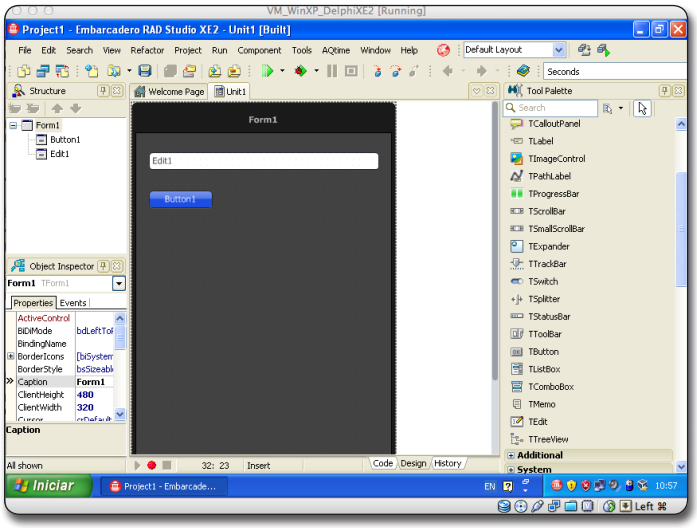

Figura 4. Adicionando componentes no formulário do iPhone

Podemos então implementar como exemplo um código bastante simples, apenas com a finalidade de testar a plataforma e demonstrar a funcionalidade, sendo assim, no evento OnClick do botão adicionamos o código da **Listagem 1**.

Listagem 1. Eventon OnClick do botão

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 edit1.Text := 'DelphiXE2 Rocks!';
end;
```

Nota-se que ao clicar no botão, uma mensagem será atribuída ao Edit. Ao executarmos a aplicação no Delphi XE2, a simulação apresenta o formulário e assim, é possível testar a aplicação, conforme mostrado na **Figura 5**.

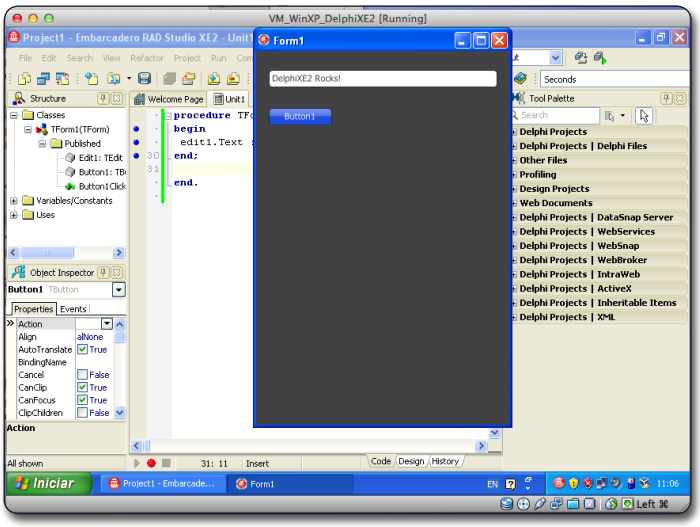

Figura 5. Simulando aplicação iOS no DelphiXE2

Agora que a aplicação está funcional, devemos salvar o projeto no diretório compartilhado. Salve-o em uma subpasta do diretório compartilhado anteriormente. Após isso compile a aplicação novamente.

Agora que o projeto está salvo e compilado devemos exportá-lo para que seja realizada a compilação no xCode. Para isso, vamos antes configurar o Delphi XE2 para realizar esta tarefa de exportação dos arquivos clicando no menu principal do Delphi XE2 na opção Tools>Configure Tools e adicionando a opção "dpr2xCode" com a propriedade "Parameters = $PROJECT", conforme mostrado na **Figura 6**.

Ao adicionarmos esta opção passaremos a contar com uma opção no menu principal do Delphi XE2 disponível em Tools>dpr2xCode, que exporta os arquivos do projeto para o formato do xCode, deixando-os prontos para serem compilados no MacOS. Ao clicar em dpr2xCode será criada uma pasta "xcode" onde o projeto foi salvo, conforme mostrado na **Figura 7**.

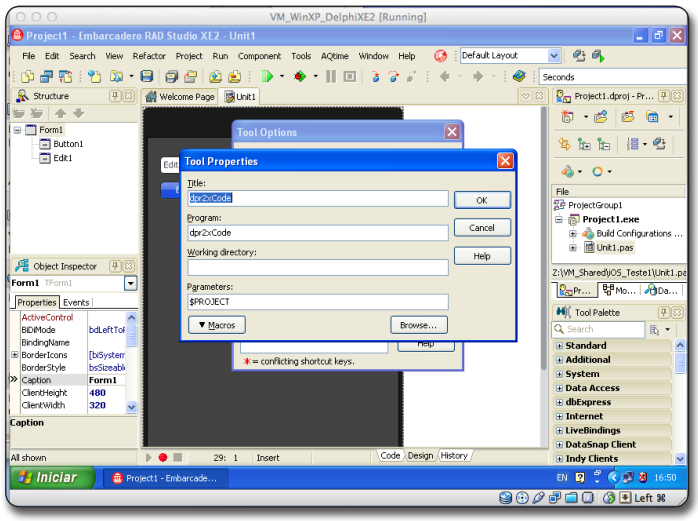

Figura 6. Preparando o menu do DelphiXE2 para exportar o projeto para xCode

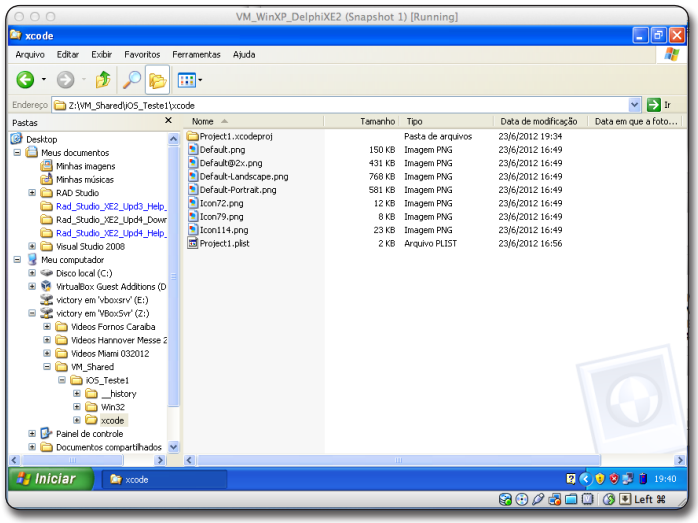

Figura 7. Pasta xcode criada no diretório do projeto após executar dpr2xcode

Após concluir os passos do xCode, precisamos instalar no MAC o arquivo FireMonkey-iOS.dmg que encontra-se no lado da VM Windows no diretório de instalação do Delphi XE2 (C:\Arquivos de programas\Embarcadero\RAD Studio\9.0\FireMonkey-iOS).

A partir da VM, copie o arquivo FireMonkey-iOS.dmg e cole no diretório compartilhado. Volte para o MacOs e execute o arquivo para iniciar a instalação do mesmo, conforme mostrado na **Figura 8**.

Para executar a instalação você deve dar um duplo clique na primeira caixa da esquerda onde se lê "fpc-2.6.0.intel-macosx.pkg". Este pacote é referente a instalação do compilador Free Pascal.

Em seguida, dê um duplo clique na segunda caixa da esquerda onde se lê "FireMonkey-iOS-XE2.pkg". Este pacote é referente à instalação do FireMonkey. A seguir, basta continuar confirmando os passos da instalação. Apesar de indicar que será concluída em menos de um minuto, a instalação deste pacote pode tomar certo tempo.

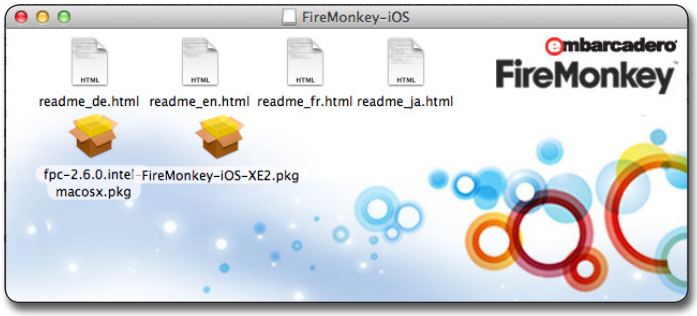

Figura 8. Instalação no MacOS do arquivo FireMonkey-iOS.dmg que acompanha o DelphiXE2

Uma vez instalados os pacotes necessários, devemos abrir o xCode através do LaunchPad do MacOS e no menu principal clicar no botão "Open", selecionando o arquivo de projeto gerado pelo processo dpr2xcode que estará na pasta do projeto, dentro do diretório compartilhado, em uma pasta chamada "xcode" conforme descrito anteriormente e mostrado na **Figura 9**.

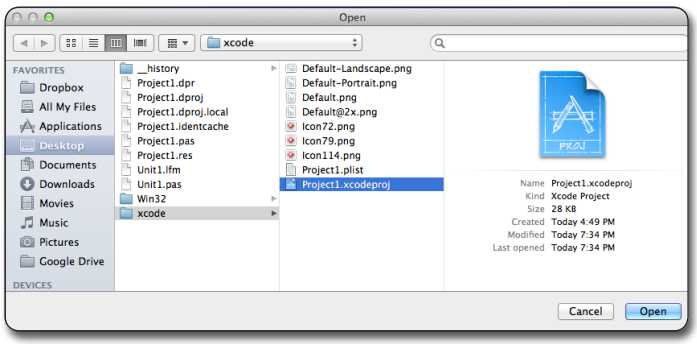

Figura 9. Abrindo projeto no xCode

Após carregar o projeto no xCode, devemos selecionar a opção iPhone 5.0 Simulator na caixa de seleção no canto superior esquerdo e em seguida clicar no botão "Play" para que a aplicação seja compilada e executada, resultando na apresentação do aplicativo desenvolvido no Delphi XE2, desta vez, rodando no simulador oficial do xCode, conforme pode ser visto na **Figura 10**.

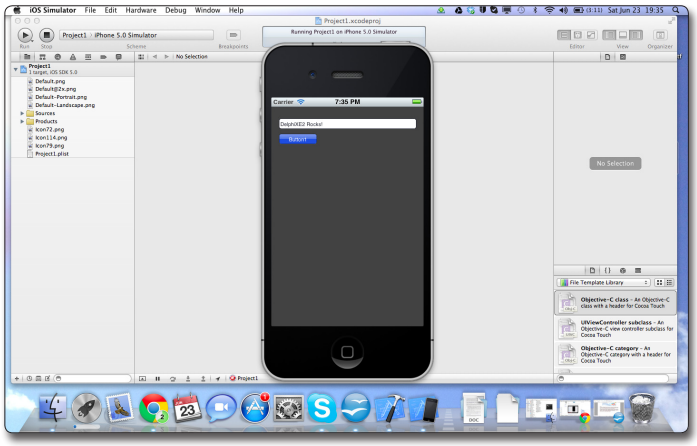

Figura 10. Aplicação criada no DelphiXE2 devidamente compilada no xCode

Missão cumprida! Concluímos todas as instalações e configurações necessárias para criar aplicações no Delphi XE2 em uma VM Windows instalada em um computador MAC. Dessa forma temos tudo pronto para começar a desenvolver nossos apps para iOS!

Instalando paleta de componentes iOS

A Embarcadero não só disponibilizou demos de aplicativos que mostram como realizar acesso a diversos recursos do iOS no iPhone e iPad como também forneceu um conjunto de componentes que simplificam bastante esta tarefa. Realize o download dos componentes disponibilizados no site da Embarcadero (seção **Links**), a pasta completa contendo os demos e os arquivos de instalação dos componentes estará disponível em sua máquina conforme mostrado na **Figura 11**.

O conteúdo de cada pasta é descrito na **Tabela 1**.

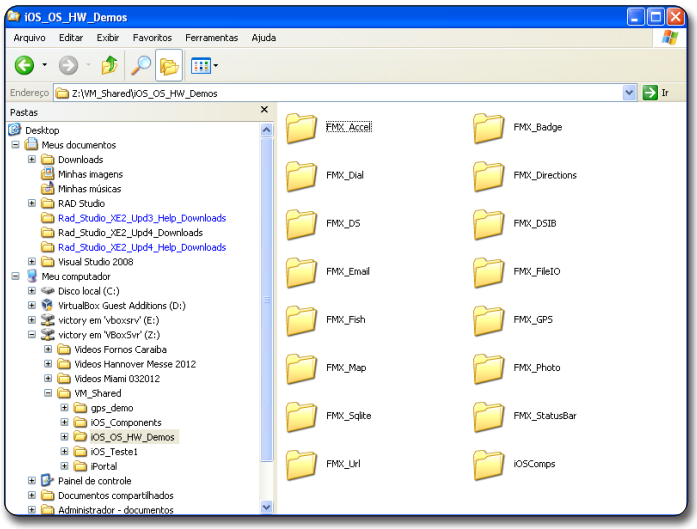

Figura 11. Pasta contendo aplicativos de demonstração e instalação dos componentes de acesso ao hardware

Nome da pasta	Descrição
FMX_Accel	Demo de leitura dos dados do acelerômetro.
FMX_Badge	Demo para controle dos números ao lado dos ícones do app.
FMX_Dial	Aplicativo para discagem de números.
FMX_Directions	Geração de rotas entre dois pontos.
FMX_DS	Demo com recursos de DataSnap.
FMX_DSIB	Demo com recursos de DataSnap com Interbase.
FMX_Email	Aplicativo de envio de e-mails.
FMX_FileIO	Aplicativo de acesso a arquivos no iOS.
FMX_Fish	FishFact no iOS.
FMX_GPS	Leitura dos dados do GPS.
FMX_Map	Acesso a mapas.
FMX_Photo	Acesso aos recursos da câmera.
FMX_Sqlite	Criação e acesso a bases de dados locais.
FMX_StatusBar	Controle da barra de status do iOS.
FMX_Url	Acesso a links de internet.
iOS_Comps	Arquivos para instalação dos componentes no DelphiXE2

Tabela 1. Conteúdo do pacote Embarcadero

Antes de abrir os demos é preciso realizar a instalação dos componentes contidos na pasta iOS_Comps. Para isso, é indicado que após realizar o download do zip, descompacte-o em um diretório "componentes" criado na pasta de instalação do Delphi XE2. O diretório final onde os componentes serão instalados será algo do tipo "C:\Arquivos de programas\Embarcadero\RAD Studio\9.0\Componentes\iOS_Components\iOSComps".

Após descompactar os arquivos, execute o Delphi XE2 e no menu principal clique em File>Open, navegue até a pasta onde os arquivos foram descompactados e selecione o arquivo iOSComps.dpk que se encontra dentro da pasta iOS_Comps.

Na janela do Project Manager clique com o botão direito sobre iOSComps.bpl e selecione a opção Build para que o projeto seja gerado. Depois de concluído, é preciso instalar a bpl, para isso clica-se novamente com o botão direito sobre o iOSComps.bpl e selecione a opção Install. É necessário adicionar o diretório dos componentes ao Library Path do Delphi.

Selecione no menu as opções Tools>Options> Library>Library Path e adicione o caminho do diretório que será algo do tipo "$(BDS)\Componentes\iOS_Components\iOSComps". Clique em OK para fechar a janela e no menu principal selecione a opção File>Close All para concluir o processo de instalação dos componentes.

Uma vez instalados é possível criar uma nova aplicação que utilize os componentes, para isso, no menu principal clique em File>New>Other>Delphi Projects>Firemonkey HD iOS Application. Surgirá um formulário com o layout do iPhone e na aba Tool Palette você encontrará os componentes instalados disponíveis no grupo iOS, conforme mostrado na **Figura 12**.

Sempre ao iniciar um novo projeto de aplicativo iOS no DelphiXE2 certifique-se de incluir as diretivas de compilação (mostradas na **Figura 13**) no início da Unit de cada formulário, caso contrário, sua aplicação poderá não executar corretamente no seu dispositivo móvel. O sintoma mais comum nestes casos é a aplicação abrir e fechar automaticamente sempre que seu ícone for clicado no aparelho. Essas diretivas indicam que, se o aplicativo for compilado pelo compilador FreePascal, que seja gerado código em Objective C.

Discando números no iPhone

Para começar podemos incluir na tela um componente do tipo TiOSPhone, que permite realizar ligações no iPhone. Incluiremos também um TEdit e um TButton. No evento OnClick do botão implemente o código mostrado na **Listagem 2**.

O número a ser discado deve ser passado para uma propriedade do tipo string e a execução do método Dial faz com que seja chamada a tela nativa do discador no iPhone. Para testar a funcionalidade desta aplicação teremos de fazer seu Deploy para o aparelho, processo que abordaremos mais adiante.

Enviando emails no iPhone e iPad

Daremos prosseguimento na nossa aplicação e incluiremos na tela um componente do tipo TiOSEmail que permite enviar e-mails no iPhone e iPad, além de outros três componentes do tipo TEdit e um TButton. No evento OnClick do botão implemente o código mostrado na **Listagem 3**.

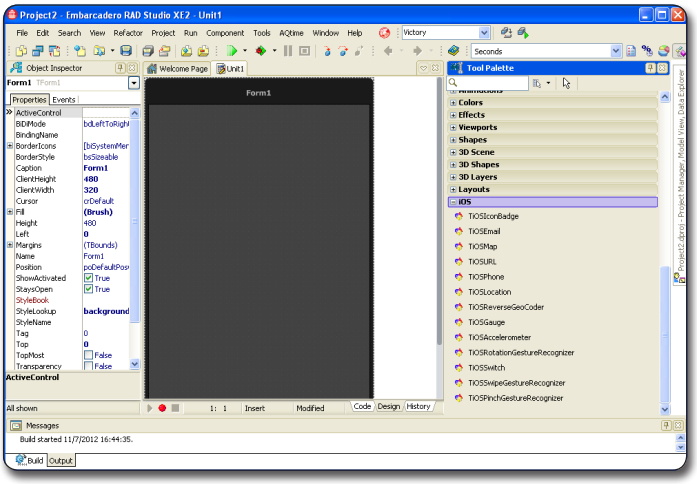

Figura 12. Lista de componentes de acesso a hardware instalados no DelphiXE2

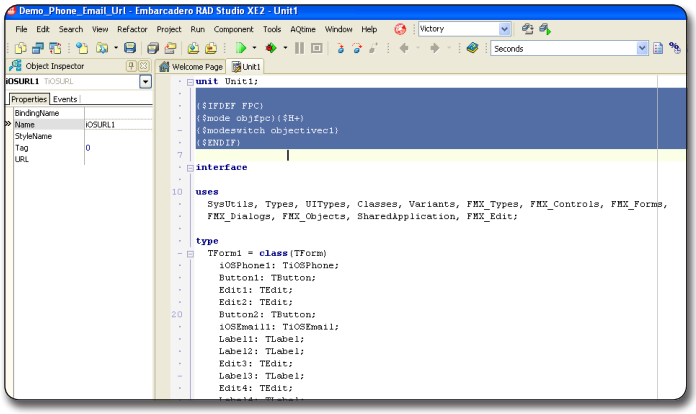

Figura 13. Diretivas de compilação necessárias as projetos iOS no DelphiXE2

Listagem 2. Evento OnClick do botão para discagem

```
01 procedure TForm1.Button1Click(Sender: TObject);
02 begin
03  iOSPhone1.Number := edit1.Text;
04  iOSPhone1.Dial;
05 end;
```

Listagem 3. Evento OnClick do botão de envio de e-mail

```
01 procedure TForm1.Button2Click(Sender: TObject);
02 begin
03  iOSEmail1.Recipient  := edit2.Text;
04  iOSEmail1.Subject    := edit3.Text;
05  iOSEmail1.Body       := edit4.Text;
06  iOSEmail1.Send;
07 end;
```

Nota

Os simuladores, a exemplo do emulador que acompanha o Delphi XE2 e/ou do xCode são muito bons e permitem testar a interface visual com o usuário de forma bastante satisfatória, no entanto, não permitem realizar testes da sua aplicação quanto ao uso de recursos específicos de hardware, a exemplo do GPS, acelerômetro, câmera bem como a recursos nativos, como o discador.

Basicamente o e-mail do destinatário, o assunto e o conteúdo da mensagem a ser enviada devem ser passados para uma propriedade do tipo String. Após isso é executado o método Send, que faz com que seja chamada a tela nativa do envio de e-mails do iPhone e/ou iPad.

Acessando URLs no iPhone e iPad

Agora incluiremos na tela um componente do tipo TiOSURL que permite acessar URLs (Uniform Resource Locators) no iPhone/iPad, adicionaremos também um TEdit e um TButton. No evento OnClick do botão implemente o código da **Listagem 4.**

Listagem 4. Evento OnClick do botão de URLs

```
01 procedure TForm1.Button3Click(Sender: TObject);
02 begin
03  iOSURL1.URL := edit5.Text;
04  iOSURL1.Open;
05 end;
```

A URL a ser acessada deve ser passada para propriedade URL a partir do texto informado no Edit. Já a chamada do método Open faz com que seja aberto o navegador nativo do iPhone/iPad. Lembre-se de sempre digitar o endereço completo, por exemplo: http://www.igara.com.br/, caso contrário, endereços como "www.igara.com.br" não executarão o comando corretamente. Após compilar e executar a aplicação no Delphi XE2 o resultado final será algo conforme mostrado a seguir na **Figura 14**.

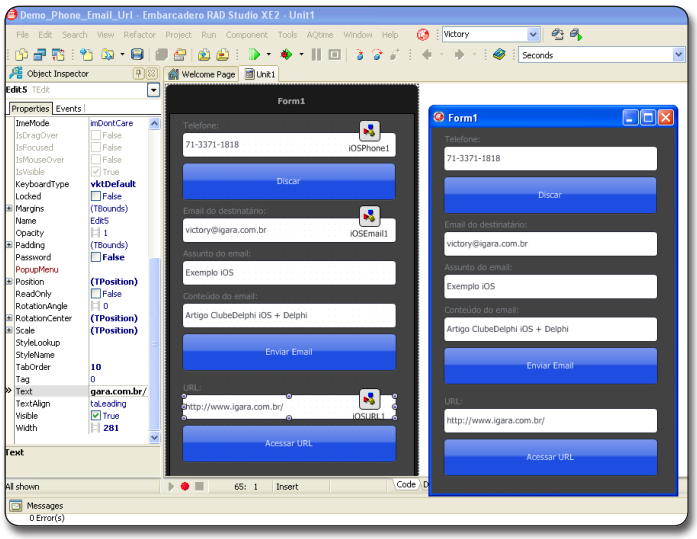

Figura 14. Aplicativo de demonstração para captura discagem, envio de emails e acesso a URLs

Lembre-se de salvar o projeto e em seguida realizar a compilação do aplicativo no Delphi XE2. Ao executá-lo, surgirá na tela a interface do iPhone gerada pelo Delphi XE2, no entanto, o emulador não simula alguns recursos, logo não teremos como avaliar a funcionalidade desta aplicação por enquanto.

Após compilar e executar aplicativo no XE2, selecione no menu principal as opções disponíveis em Tools>dpr2xCode para exportar o projeto para xCode e deixá-lo pronto para compilação no OS X.

Compilando a aplicação completa no xCode

No menu principal do xCode, clique no botão Open e selecione o arquivo de projeto com extensão ".xcodeproj" gerado pelo processo dpr2xcode. Após carregar o projeto, devemos selecionar a opção "iPhone 5.0 Simulator" e em seguida clicar no menu Product>Build para que a aplicação seja compilada. Se tudo foi feito corretamente até aqui o resultado da **Figura 15** poderá ser visto.

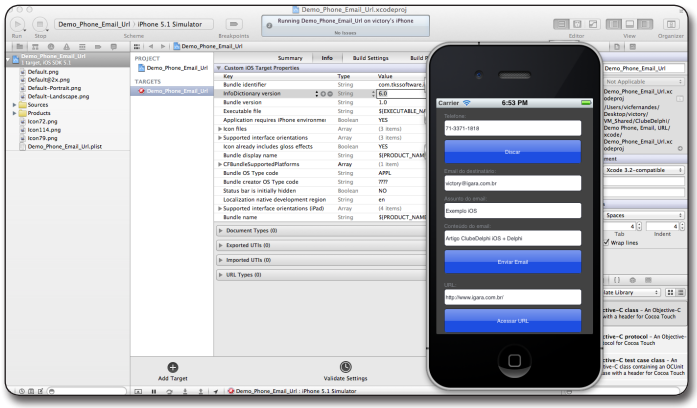

Figura 15. Aplicativo demo compilado no xCode e rodando no ambiente de simulação

Agora que a aplicação está compilando e executando tanto no Delphi XE2 quanto no xCode, estamos prontos para sair do ambiente de simulação e finalmente carregar o aplicativo nos dispositivos.

Carregando a aplicação no iPhone ou iPad

Para utilizar suas próprias aplicações nos dispositivos é preciso fazer parte do iOS Developer Program (seção **Links**). O ingresso nesse programa depende do preenchimento de formulários específicos e pagamento de uma taxa anual no valor de US$ 99.

Após se inscrever no programa e receber os e-mails de confirmação da Apple você deverá conectar o seu aparelho na porta USB do seu MAC e clicar no botão Organizer localizado no canto direito superior da tela principal do xCode, conforme a **Figura 16**.

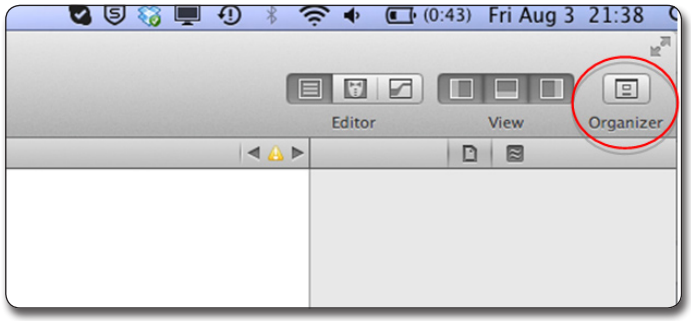

Figura 16. Botão Organizer na tela do xCode

Em seguida, localize o dispositivo conectado na lista ao lado esquerdo e clique no botão "Use for Development" (**Figura 17**).

Uma vez adicionado, o seu dispositivo será listado no canto esquerdo superior da tela do xCode, conforme mostrado na **Figura 18** e você poderá selecioná-lo de forma que, após realizar a compilação, o aplicativo não será mais executado no compilador mas sim diretamente no aparelho.

Figura 17. Adicionando o seu dispositivo no xCode para uso em desenvolvimento

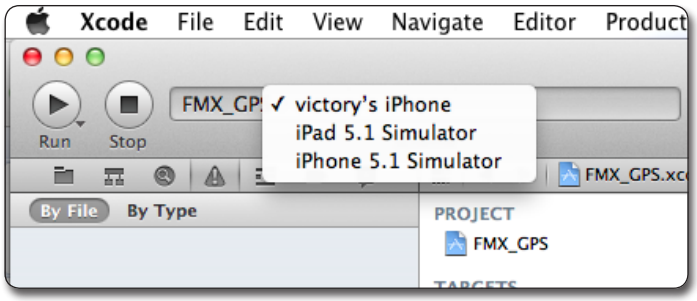

Figura 18. Selecionando no xCode o ambiente/dispositivo onde será realizada a execução do aplicativo

É necessário gerar os chamados arquivos de "provisioning profile" no site da Apple para cada aplicativo criado para só então executá-los em um dispositivo. Para gerar este arquivos de provisioning profile, é preciso acessar o portal de desenvolvedores da Apple (seção **Links**) e clicar nas opções "iOS Dev Center>iOS Provisioning Portal>App IDs>New App ID". Preencha o campo Description do formulário com o nome da sua aplicação e o campo Bundle Identifier, conforme instruções contidas no próprio site, com o nome reverso do seu domínio concatenado com o nome do seu aplicativo, resultando em algo conforme mostrado na **Figura 19.**

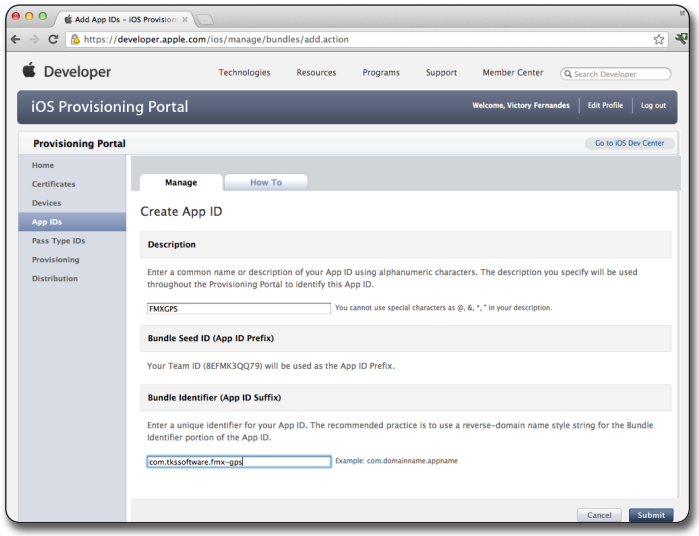

Figura 19. Criando App IDs no portal de Provisioning Profile da Apple

Neste exemplo, utilizamos como referência o título do certificado usado para criação de uma aplicação GPS entitulada "FMX_GPS". No xCode, o mesmo certificado pode ser compartilhado entre aplicações diferentes independente do nome do projeto, no entanto, o ícone correspondente à sua aplicação na tela do seu iPhone e/ou iPad será sempre substituído pelo novo aplicativo que utilizar o certificado compartilhado durante a gravação.

Após criar o App ID, clique no link Provisioning do lado esquerdo da tela e em seguida Development>New Profile. Preencha os campos do formulário, onde o nome do Profile tem o mesmo nome da aplicação. No campo App ID selecionamos o App ID criado no passo anterior e na opção Device informamos nosso dispositivo que foi adicionado ao portal quando anteriormente o informamos no Organizer do xCode.

Com o Development Profile criado, atualize a tela para visualizar a lista dos profiles disponíveis e realize o download do profile recém-criado clicando no botão "download".

Agora clique na aba Distribution e em seguida no botão New Profile. Preencha os campos do formulário, onde selecionamos o Distribution Method como sendo da opção Ad Hoc, uma vez que não estaremos inicialmente distribuindo o aplicativo na App Store, mas sim, realizando testes em um dispositivo plugado ao computador. O campo Profile Name deve ter um nome diferente do nome dado ao Development Profile, neste caso foi adicionada uma letra "d" ao final, e por fim, selecionamos o App ID criado anteriormente e marcamos o dispositivo no qual o aplicativo será testado. É possível cadastrar até 100 dispositivos para testes. Após criar o Distribution Profile, atualize a tela para visualizar a lista dos profiles disponíveis e realize o download do profile recém-criado, clicando no botão "download". Ao fazer o download dê um duplo clique nos arquivos de Development e Distribution Profile para adicioná-los ao xCode que listará os mesmos na tela de Organizer.

Execute novamente a compilação e caso ainda seja disparado o erro referente ao Provisioning Profile, certifique se o valor da

propriedade Bundle Identifier é idêntico ao valor utilizado na criação do App ID, conforme mostrado na **Figura 20**.

Missão cumprida! Se todos os passos foram executados corretamente você deve ser capaz de realizar a compilação do seu aplicativo e o mesmo será carregado e executado no dispositivo. Na **Figura 21** é possível vê-lo executando no simulador do xCode e no iPhone 4s.

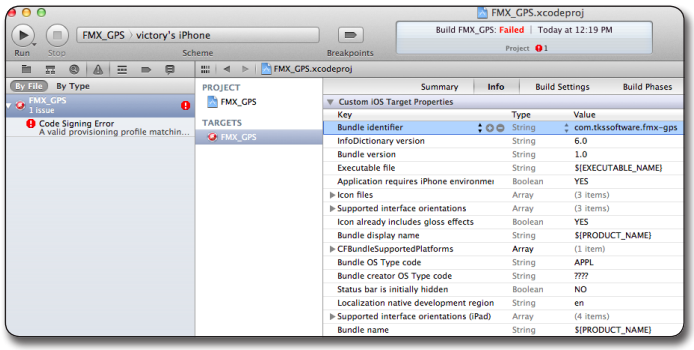

Figura 20. Propriedade Bundle Identifier do projeto no xCode deve ser igual ao valor atribuido durante a criação do App ID

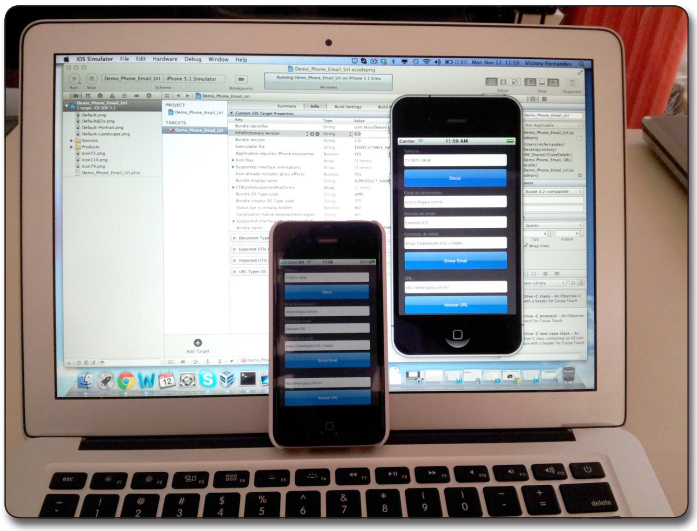

Figura 21. Aplicativo compilado no xCode e executndo no Iphone 4S operando iOS versão 5.1

Conclusão

Com este artigo demonstramos todas as etapas necessárias para que seja possível iniciar suas atividades de desenvolvimento de apps para iPhone e iPad utilizando o Delphi XE2. Cobrimos a instalação da VM, configuração do ambiente no MAC, configurações no Delphi que permitem exportar seus projetos para o xCode, o processo de compilação e execução no ambiente de simulação do MAC e por último, a aplicação rodando direto nos dispositivos. Conhecemos um pouco da parte burocrática do desenvolvimento, que envolve o iOS Developer Program, além das configurações específicas para geração de Provisioning Profiles para cada app gerado, de forma a ser capaz de carregá-los nos dispositivos.

As etapas demonstradas podem ser seguidas, para que a partir desta leitura, você possa dar continuidade aos seus estudos realizando a compilação dos demais aplicativos de demonstração fornecidos pela Embarcadero, que incluem acesso aos recursos de acelerômetro, GPS, bússola, dentre outros.

Sem dúvida, o Delphi XE2 é uma excelente plataforma para desenvolvimento de aplicativos para iOS.

Victory Fernandes
Embarcadero Most Valuable Professional – MVP Brasil. Professor do Departamento de Engenharia da Faculdade Area1 e Diretor técnico da TKS Software.

Aplicativos disponibilizados por Anders Ohlsson no site da Embarcadero
http://cc.embarcadero.com/item/28635.

Download do xCode para Mac
https://developer.apple.com/xcode/

Site do VMWare
www.vmware.com

Site do Parallels
www.parallels.com

Virtual Box
www.virtualbox.org

Download do xCode
https://developer.apple.com/xcode/

Release Note oficial Embarcadero
http://docwiki.embarcadero.com/RADStudio/en/Release_Notes_for_XE2_Update_4#2._Reinstall_FireMonkey_iOS_Tools

Download dos componentes da Embarcadero para desenvolvimento móvel
http://cc.embarcadero.com/item/28635

iOS Developer Program
https://developer.apple.com/programs/

Portal de desenvolvedores Apple
https://developer.apple.com/

Adobe Phonegap Build
https://build.phonegap.com/

Dê seu feedback sobre esta edição!

A ClubeDelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/clubedelphi/feedback

Editor de Etiquetas com FastReport

Como criar etiquetas usando Delphi XE2 e FastReport

Um dos maiores trabalhos dos desenvolvedores sem dúvida alguma está relacionado à parte de relatórios. Grande parte do desenvolvimento de um Software, independente do segmento, é a confecção de relatórios para clientes. Esta geralmente é uma tarefa complicada, na qual exige certo tempo do desenvolvedor apenas para a apresentação de dados da melhor forma possível. Entre estas tarefas, é necessário escrever comandos SQL, configuração de componentes de conexão, criação de telas com parâmetros, layout do relatório, configuração de margens, espaçamento, tipo de fonte e uma dezena de outras opções que vão exigir mais ou menos recursos de acordo com o que o cliente deseja.

Neste processo de relatórios, quem desenvolve há muito tempo (desde as versões mais antigas do Delphi) sabe que o processo de transição de relatórios na IDE já passou por várias suítes. Dentre as mais conhecidas podemos citar o Quick Report e Rave Report (que acompanhavam o Delphi) e outros também conhecidos como o Report Builder e Fortes Reports. Pois bem, agora o Delphi conta com um excelente e poderoso aliado para a criação de relatórios da mais ampla gama de possibilidades, o Fast Report, que agora acompanha a instalação do Delphi XE2.

Para quem está conhecendo agora este poderoso gerador de relatórios, o Fast Report conta com inúmeros recursos presentes nos demais (além de vários inovadores). Entre estes podemos citar a possibilidade de criar servidores de relatórios (através da versão Enterprise), relatórios na Web, criação de diálogos para entrada de parâmetros (de forma automática), entre outras. Destaque também por ser a primeira suíte de relatórios a suportar o FireMonkey.

Embora a gama de suíte de relatórios seja imensa, o desejo de todo desenvolvedor é transferir a possibilidade da confecção de relatórios para o suporte ou cliente, porém, aqui existem algumas questões que esbarram nesta possibilidade. Algumas que podemos citar são:

Resumo DevMan

Editor de Etiquetas com FastReport:

O FastReport é um gerador de relatórios que garante muita flexibilidade e reutilização, pois seu trabalho é baseado em arquivos. Assim, é praticamente possível criar vários modelos, ou alterar a estrutura de um modelo em tempo de execução e conseguir um novo layout de impressão. Com isso em mente, nesse artigo iremos aprender a fazer um aplicativo que permitirá a um usuário criar o seu modelo de etiqueta, da forma que ele precisa.

Em que situação o tema é útil:

Criar um editor de etiquetas irá, além de agregar valor ao seu produto, dar total liberdade ao seu cliente quanto ao layout de impressão. A quantidade de pedidos para alteração de algum campo, ou novo modelo sempre é relativamente alta, então por que não criar um editor onde o cliente poderá criar, editar com total liberdade? Vamos então colocar esta ideia em prática.

• Falta de conhecimento técnico: embora a criação da maioria dos relatórios seja relativamente simples, ainda assim este processo exige um conhecimento técnico sobre propriedades do relatório, disposição, configuração, conexão com dados e etc.
• Conhecimento de ANSI SQL: este pode ser considerado o maior problema, já que em muitos casos, é necessário não só o conhecimento técnico, mas sim o conhecimento sobre os relacionamentos entre as tabelas e até mesmo questões que vão além, como performance.

Deixando um pouco de lado estas questões, poderíamos imaginar um cenário aonde qualquer pessoa facilmente confeccionaria etiquetas ou relatórios. É exatamente este o propósito deste artigo. Criaremos uma solução simples de impressão de etiquetas que apresenta sugestões de como você pode elaborar recursos em sua aplicação para que seja possível repassar a responsabilidade de montagem do relatório (design e impressão) para a sua equipe de suporte, ou usuário final, de uma forma simples e objetiva,

utilizando para isso recursos do Fast Reports e do próprio Delphi. Deste modo, aquelas situações de mudança de layout para alterações simples, como etiquetas, não são mais problema para o desenvolvedor.

Como o FastReport utiliza um arquivo separado para armazenar o layout de impressão, é possível customizar um mesmo relatório para vários formatos. Quando aqui digo relatório, digo informações a serem impressas. O relatório final é a saída deste processo.

O aplicativo

Para o nosso exemplo iremos utilizar Delphi XE2, FastReport 4 e Firebird. A intensão de usar o FastReport é aproveitar todas as suas ferramentas internas para a construção da etiqueta. Todas as características de layout como altura, largura, tamanho de página e margens serão gravados em uma tabela de um banco de dados que criaremos a seguir. Assim, faremos um novo aplicativo.

Criando o banco de dados

Crie um banco de dados chamado Etiquetas.fdb e utilize o gerenciador de sua preferência. Sugiro o IBExpert por ser bem simples de manusear. As tabelas a serem criadas e algumas informações de preenchimento podem ser vistas na **Listagem 1**.

A tabela CLIENTE é nossa fonte de dados. Nas etiquetas constarão suas informações. Já na tabela ETIQUETA, armazenamos a estrutura do layout. Ao observar pela **Listagem 1** o nome dos campos, vemos que é possível atingir o posicionamento completo não só de uma etiqueta, mas de seu conteúdo também. Temos o posicionamento de cada campo a ser impresso.

Conectando ao banco de dados

Para fazermos a conexão com o banco iremos utilizar a suíte DBX (dbExpress) com ClientDataSet. Foi adicionado ao formulário principal um SQLConnection, dois ClientDataSet, dois DataSetProvider e dois SQLQuery. Cada um foi renomeado conforme mostrado na **Figura 1**.

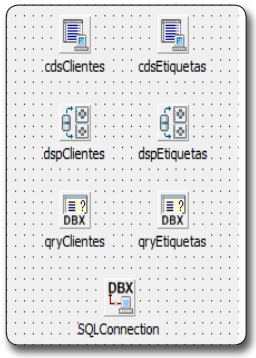

Figura 1. Componentes de conexão com o banco de dados

Realizando a configuração dos componentes, no SQLConnection selecionamos a propriedade ConnectionName e mudamos seu valor para FBConnection, já na propriedade Params se informa o local em que o banco foi criado no início.

Nas Querys é necessário informar qual o componente responsável pela conexão com o banco de dados, sendo assim, deve-se ligar o SQLConnection à propriedade de mesmo nome das Querys. Já no DatasetProvider, suas propriedades DataSet devem ser ligadas as respectivas Queries e, por último, no ClientDataSet, o Provider receberá o DataSetProvider.

Como as etiquetas exibirão os dados dos Clientes, o componente qryClientes teve seu SQL ajustado para a sentença a seguir, como mostra a **Listagem 2**. Ao executar essa instrução, todo o conteúdo da tabela Cliente será recuperado e, através do DataSetProvider, automaticamente será carregado para o cdsClientes, permanecendo em memória e à disposição do relatório em criação.

Listagem 1. Script do banco de dados

```
CREATE TABLE CLIENTE (
  CLI_ID INTEGER NOT NULL,
  CLI_NOME VARCHAR(50) CHARACTER SET WIN1252 COLLATE WIN_PTBR,
  CLI_TELEFONE VARCHAR(10));

INSERT INTO CLIENTE (CLI_ID, CLI_NOME, CLI_TELEFONE)
VALUES (1, 'RAFAEL BRENDOLAN', '1655551111');

INSERT INTO CLIENTE (CLI_ID, CLI_NOME, CLI_TELEFONE)
VALUES (2, 'JOSE ROBERTO', '1655552222')

INSERT INTO CLIENTE (CLI_ID, CLI_NOME, CLI_TELEFONE)
VALUES (3, 'LIDIANE SILVA', '1655553333');

INSERT INTO CLIENTE (CLI_ID, CLI_NOME, CLI_TELEFONE)
VALUES (4, 'JOAO ALFREDO', '1655554444');

INSERT INTO CLIENTE (CLI_ID, CLI_NOME, CLI_TELEFONE)
VALUES (5, 'PAULO PEDRO', '1655556666');

CREATE TABLE ETIQUETA (
  ID_ETIQUETA  INTEGER NOT NULL,
  ALTURA_ETIQUETA  DOUBLE PRECISION,
  LARGURA_ETIQUETA  DOUBLE PRECISION,
  LARGURA_COLUNA_ETIQUETA  DOUBLE PRECISION,
  DISTANCIA_HORIZONTAL  DOUBLE PRECISION,
  DISTANCIA_VERTICAL  DOUBLE PRECISION,
  NUM_COLUNAS  INTEGER,
  MARGEM_ESQUERDA_PAG  DOUBLE PRECISION,
  MARGEM_DIREITA_PAG  DOUBLE PRECISION,
  MARGEM_SUP_PAG  DOUBLE PRECISION,
  MARGEM_INF_PAG  DOUBLE PRECISION,
  ALTURA_PAG  DOUBLE PRECISION,
  LARGURA_PAG  DOUBLE PRECISION,
  LARGURA_CAMPO_NOME  DOUBLE PRECISION,
  ALTURA_CAMPO_NOME  DOUBLE PRECISION,
  TOP_CAMPO_NOME  DOUBLE PRECISION,
  LEFT_CAMPO_NOME  DOUBLE PRECISION,
  LARGURA_CAMPO_FONE  DOUBLE PRECISION,
  ALTURA_CAMPO_FONE  DOUBLE PRECISION,
  TOP_CAMPO_FONE  DOUBLE PRECISION,
  LEFT_CAMPO_FONE  DOUBLE PRECISION
);
```

O mesmo é feito para a tabela ETIQUETA. O SQL da **Listagem 3** é inserido no componente qryEtiquetas e assim, o procedimento será semelhante ao que já ocorre com a tabela CLIENTE.

Após a definição do SQL, os campos persistentes devem ser adicionados (clique com botão direito do mouse sobre o SQLQuery e ClientDataSet e escolha Add all fields). Ao inserir os campos é possível acessá-los via código, como se fossem objetos comuns, assim é possível ajustar suas propriedades e obter/passar valor a eles.

Listagem 2. SQL de busca de clientes

```
SELECT
  CLI_ID,
  CLI_NOME,
  CLI_TELEFONE
FROM
  CLIENTE
```

Listagem 3. SQL de busca de etiquetas

```
SELECT
  ID_ETIQUETA,
  ALTURA_ETIQUETA,
  LARGURA_ETIQUETA,
  LARGURA_COLUNA_ETIQUETA,
  DISTANCIA_HORIZONTAL,
  DISTANCIA_VERTICAL,
  NUM_COLUNAS,
  NUM_LINHAS,
  MARGEM_ESQUERDA_PAG,
  MARGEM_DIREITA_PAG,
  MARGEM_SUP_PAG,
  MARGEM_INF_PAG,
  ALTURA_PAG,
  LARGURA_PAG,
  TIPO_PAG,
  LARGURA_CAMPO_NOME,
  ALTURA_CAMPO_NOME,
  TOP_CAMPO_NOME,
  LEFT_CAMPO_NOME,
  LARGURA_CAMPO_FONE,
  ALTURA_CAMPO_FONE,
  TOP_CAMPO_FONE,
  LEFT_CAMPO_FONE
FROM
  ETIQUETA
```

Criando os campos de configurações

Ao formulário principal do projeto adicionamos sete objetos Button. Cada um deve ter sua propriedade Caption alterada para o seguinte: Adicionar, Editar, Anterior, Próximo, Remover, Salvar Modelo e Visualizar. No evento OnClick de cada um deles é necessária uma codificação usando o ClientDataSet, como mostrado na **Tabela 1.**

Temos operações básicas de inserção, edição e navegação. No botão Visualizar executamos o método ShowReport de um objeto FrxReport, que ainda não foi inserido no formulário. Esse método realiza a pré-visualização de um relatório.

Temos também a chamada de dois métodos, SalvarModelo e CarregarModelo, que serão abordados posteriormente.

Para a criação dos campos de edição que irão conter os dados do layout, simplesmente efetuamos um clique duplo sobre o ClientDataSet CdsEtiquetas, assim a janela do Fields Editor surge e podemos arrastar para o formulário cada um dos Fields. Na **Tabela 2** temos os campos que devem ser arrastados e como deve ser ajustada a propriedade Name de cada componente TDBEdit que é criado pelo IDE.

Botão	Código
Adicionar	CdsEtiquetas.Insert;
Editar	CdsEtiquetas.Edit;
Anterior	CdsEtiquetas.Prior; CarregarModelo;
Próximo	CdsEtiquetas.Next; CarregarModelo;
Remover	CdsEtiquetas.Delete; CdsEtiquetas.ApplyUpdates(0);
Salvar Modelo	SalvarModelo; CarregarModelo;
Visualizar	FrxReport.ShowReport(true);

Tabela 1. Evento OnClick dos botões

Campo	Propriedade Name do respectivo TDBEdit
ID_ETIQUETA	DBEdit1
ALTURA_ETIQUETA	edtAltura_Etiqueta
LARGURA_ETIQUETA	edtLargura_Etiqueta
LARGURA_COLUNA_ETIQUETA	edtLargura_Coluna_Etiqueta
DISTANCIA_HORIZONTAL	edtDistancia_Horizontal
DISTANCIA_VERTICAL	edtDistancia_Vertical
NUM_COLUNAS	edtNum_Colunas

Tabela 2. Campo de edição do layout

Configurando o FastReport

Partiremos agora para o processo de configuração do Fast Report. Para simular o nosso exemplo, adicione ao formulário um componente FrxReport e dois frxDBDataSets, conforme exibido na **Figura 2**.

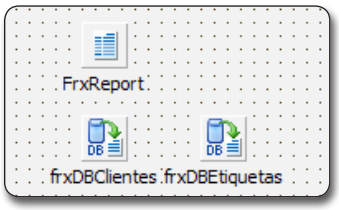

Figura 2. FrxReport e FrxDBDataSet adicionados ao projeto

Selecione o componente frxDBClientes e aponte a propriedade DataSet para o componente. cdsClientes. Faça o mesmo processo no frxDBEtiquetas, porém, selecionando o DataSet cdsEtiquetas.

Dando um duplo-clique sobre os componentes, será aberta a janela de Field Aliases, onde visualizaremos o nome dos campos importados pela nossa instrução SQL. Estes são os campos que teremos visualização dentro do Fast.

Nota

No frxReport há uma propriedade chamada "StoreInDFM". Caso marcada, o relatório será automaticamente salvo no próprio componente, conforme o padrão do Delphi para telas e componentes que armazenam suas próprias propriedades em arquivos .dfm, que servem como uma espécie de instrução. Como em nosso projeto teremos apenas um relatório e os modelos das etiquetas irão apenas mudar as suas próprias propriedades utilizaremos esta opção como default = "True".

Para agilizar os processos, no lugar de abrir o Preview do relatório para visualizar como está ficando a etiqueta durante o processo de criação, iremos criar essa visualização abaixo das propriedades da etiqueta. Para isso, utilizaremos o componente TFrxDesigner (**Figura 3**).

Figura 3. Componente FrxDesigner

Vale lembrar que não é preciso fazer nenhuma modificação no componente FrxDesigner. Para abrir em nosso projeto a visualização em Runtime, utilizaremos o evento OnCreate do formulário, conforme sugerido na **Listagem 4**.

Listagem 4. Evento OnCreate do formulário principal

```
01 procedure TFrmPrincipal.FormCreate(Sender: TObject);
02 begin
03  cdsClientes.Open;
04  cdsEtiquetas.Open;
05  FrxReport.DesignReportInPanel(Panel1);
06 end;
```

O código da **Listagem 4** simplesmente abre os componentes, exibindo os dados (linhas 03 e 04). A seguir, no método DesignReportInPanel, como o próprio nome sugere, o Designer fica contido dentro de um Container, que no caso é nosso componente Panel.

Conforme mencionado anteriormente, iremos codificar os evento OnClick dos botões que serão responsáveis pela navegação dos dados. Sendo assim, implemente-os conforme a **Listagem 5**. Note que devido a simplicidade, todos foram adicionados a uma mesma listagem.

Ao executar a aplicação, sua aparência será semelhante a da **Figura 4**.

Observando a **Figura 4**, notamos que as barras de ferramentas do FastReport estão visíveis, no entanto, não queremos esses detalhes em nossa aplicação. Antes de removê-las, faremos a configuração do nosso relatório para a criação de etiquetas. Sendo assim, clique no menu Relatório e escolha a opção Dados. Selecione os dois DataSets. A seguir, adicione uma banda de dados Mestre, como mostrado na **Figura 5**.

Feitos estes processos, será exibida uma janela para a seleção do DataSet. Como nosso objetivo é exibir os dados dos Clientes, adicionaremos o componente frxDBClientes1. Nossa "Árvore de Dados" (localizada do lado direito) deverá estar como na **Figura 6**.

Clique sobre a árvore e arraste os campos "CLI_NOME" e "CLI_TELEFONE" para dentro do Master Data adicionado anteriormente. Antes de prosseguir, vale a pena fazer algumas ressalvas sobre os campos que iremos trabalhar no Fast Report.

Listagem 5. Eventos OnClick dos botões do formulário

```
//btnAdicionar
cdsEtiquetas.Insert;
//btnEditar
cdsEtiquetas.Edit;
//btnSalvar
cdsEtiquetas.Post;
cdsEtiquetas.ApplyUpdates(0);
//btnProximo
cdsEtiquetas.Next;
//btnVoltar
cdsEtiquetas.Prior;
//btnRemover
cdsEtiquetas.Delete;
cdsEtiquetas.ApplyUpdates(0);
```

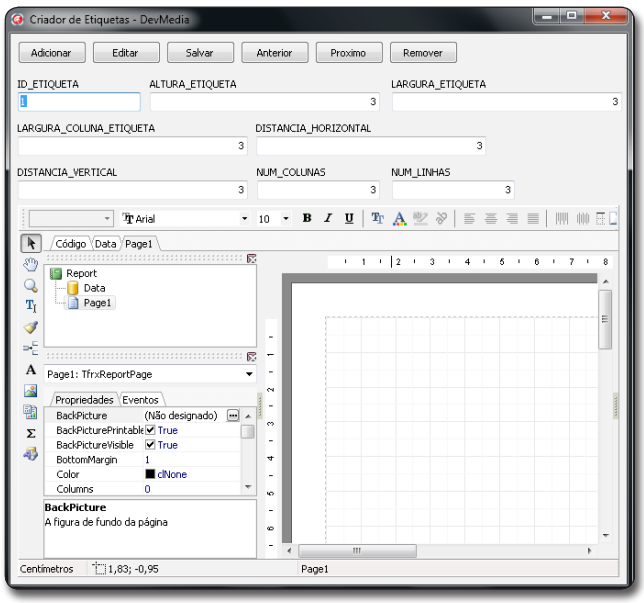

Figura 4. Projeto de Etiquetas em execução

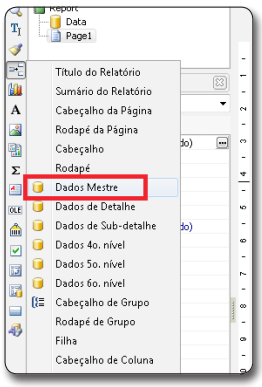

Figura 5. Inserindo a banda "MasterData"

Também no Master Data temos a propriedade Columns que se refere obviamente a quantidade de colunas. Defina seu valor com 2 colunas. Adotaremos a largura de 9 cm, sendo assim, informe em ColumnWidth o valor 9. ColumnGap é responsável por indicar o espaço horizontal ente as etiquetas, indicaremos o valor de 1 cm a ela. Para finalizar as configurações de medidas, iremos informar a altura. Assim, defina o valor 3 para a propriedade Height.

Aqui temos um possível desafio. Qual será o valor do espaço vertical entre as etiquetas? Isso pode ser resolvido de várias formas, e uma delas é a adição de uma banda filha ligada ao nosso Master Data. Assim, toda vez que for impressa uma nova linha, será adicionada uma cópia dessa banda filha abaixo dela.

Clique em Inserir Banda e selecione a opção Filha como mostra a **Figura 7**, informando seu Name como "Filha".

Após ter adicionado a banda filha, deverá ser realizada a ligação. Para isso, clique na Master Data e informe a banda recentemente criada, através da propriedade Child.

Antes de finalizarmos, serão criados os métodos restantes, responsáveis por carregar e salvar os modelos de etiquetas. Sendo assim, o primeiro método é o "SalvarModelo", exibido na **Listagem 6.**

Embora grande, a **Listagem 6** não é complexa. Como se percebe, na linha 03 é criada uma variável reportpage do tipo TfrxComponent. Esse tipo é o tipo base dos controle do Fast Report. Ela será utilizada para identificar cada um dos objetos contidos no relatório. Na linha 06 esta variável é utilizada com o método FindObject, retornando assim um ponteiro que aponta para o objeto relatório, caso encontre o objeto chamado 'Page1'. Na linha 07 é verificado se o tipo de classe contido na variável reportpage é do tipo TfrxReportPage. Caso seja, o cdsEtiqueta é preenchido através de TypeCast (**ver Nota do DevMan 1**) com as respectivas propriedades referentes a margem (linhas 09 a 14). Já na linha 16,

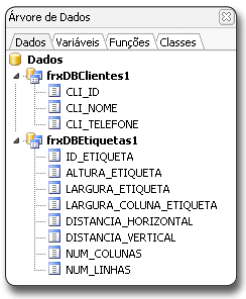

Figura 6. Árvore de Dados já configurada

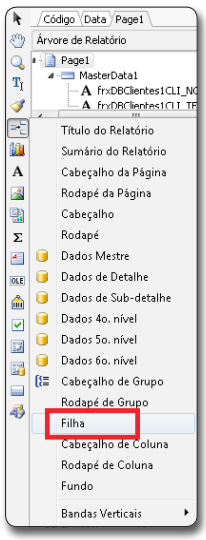

Figura 7. Adicionando uma banda Child

Listagem 6. Procedures SalvarModelo

```
01 procedure TFrmPrincipal.SalvarModelo;
02 var
03  reportpage : TfrxComponent;
04 begin
05  cdsEtiquetas.Edit;
06  reportpage := FrxReport.FindObject('Page1');
07  if (reportpage.ClassType = TfrxReportPage) then
08  begin
09   cdsEtiquetasALTURA_PAG.AsVariant :=
    (reportpage as TfrxReportPage).PaperHeight;
10   cdsEtiquetasLARGURA_PAG.AsVariant :=
    (reportpage as TfrxReportPage).PaperWidth;
11   cdsEtiquetasMARGEM_ESQUERDA_PAG.AsVariant :=
    (reportpage as TfrxReportPage).LeftMargin;
12   cdsEtiquetasMARGEM_SUP_PAG.AsVariant :=
    (reportpage as TfrxReportPage).RightMargin;
13   cdsEtiquetasMARGEM_SUP_PAG.AsVariant :=
    (reportpage as TfrxReportPage).TopMargin;
14   cdsEtiquetasMARGEM_INF_PAG.AsVariant :=
    (reportpage as TfrxReportPage).BottomMargin;
15  end;
16  reportpage := FrxReport.FindObject('MasterData1');
17  if (reportpage.ClassType = TfrxMasterData) then
18  begin
19   cdsEtiquetasNUM_COLUNAS.AsVariant := StrToFloat(edtNum_Colunas.Text);
20   cdsEtiquetasLARGURA_ETIQUETA.AsVariant :=
    StrToFloat(edtLargura_Etiqueta.Text);
21   cdsEtiquetasALTURA_ETIQUETA.AsVariant :=
    StrToFloat(edtAltura_Etiqueta.Text);
22   cdsEtiquetasDISTANCIA_HORIZONTAL.AsVariant :=
    StrToFloat(edtDistancia_Horizontal.Text);
23  end;
24  reportpage := FrxReport.FindObject('Filha');
25  if (reportpage.ClassType = TfrxChild) then
26  begin
27   cdsEtiquetasDISTANCIA_VERTICAL.AsVariant :=
    StrToFloat(edtDistancia_Vertical.Text);
28  end;
29  reportpage := frxReport.FindObject('frxDBClientes1CLI_NOME');
30  if (reportpage.ClassType = TfrxMemoView) then
31  begin
32   cdsEtiquetasALTURA_CAMPO_NOME.AsVariant :=
    (reportpage as TfrxMemoView).Height;
33   cdsEtiquetasLARGURA_CAMPO_NOME.AsVariant :=
    (reportpage as TfrxMemoView).Width;
34   cdsEtiquetasTOP_CAMPO_NOME.AsVariant :=
    (reportpage as TfrxMemoView).Top;
35   cdsEtiquetasLEFT_CAMPO_NOME.AsVariant :=
    (reportpage as TfrxMemoView).Left;
36  end;
37  reportpage := FrxReport.FindObject('frxDBClientes1CLI_TELEFONE');
38  if (reportpage.ClassType = TfrxMemoView) then
39  begin
40   cdsEtiquetasALTURA_CAMPO_FONE.AsVariant :=
    (reportpage as TfrxMemoView).Height;
41   cdsEtiquetasLARGURA_CAMPO_FONE.AsVariant :=
    (reportpage as TfrxMemoView).Width;
42   cdsEtiquetasTOP_CAMPO_FONE.AsVariant :=
    (reportpage as TfrxMemoView).Top;
43   cdsEtiquetasLEFT_CAMPO_FONE.AsVariant :=
    (reportpage as TfrxMemoView).Left;
44  end;
45  cdsEtiquetas.Post;
46  cdsEtiquetas.ApplyUpdates(0);
47  FastReportRefresh;
48 end;
```

novamente é utilizado o método FindObject para identificar se há um objeto MasterData, retornando-o para a variável ReportPage. A seguir, na linha 17 é testado se a classe é a TfrxMasterData. Caso seja, são atribuídos ao cdsEtiquetas os valores referentes a largura, altura e distância horizontal (linhas 19 a 22). Na linha 29 é pesquisado o objeto frxDBClientes1CLI_NOME, para a seguir verificar se sua classe é do tipo TfrxMemoView. Sendo assim, o mesmo processo de preencher o cdsEtiqueta é realizado, porém, dessa vez com os campos ALTURA_CAMPO_NOME, LARGURA_CAMPO_NOME, TOP_CAMPO_NOME e LEFT_CAMPO_NOME. Na linha 37 o processo é realizado pela última vez, desta vez verificando o objeto frxDBClientes1CLI_TELEFONE e a classe TfrxMemoView. Finalizando, os demais valores são atribuídos ao cdsEtiquetas (linhas 40 a 43). Após todo o processo, os dados são persistidos no banco e o método FastReportRefresh (criado mais adiante) é chamado (linhas 45 a 47).

Listagem 7. Método FastReportRefresh

```
01 procedure TFrmPrincipal.FastReportRefresh;
02 begin
03  Panel1.Repaint;
04  FrxReport.Designer.Destroy;
05  FrxReport.Designer := nil;
06  FrxReport.DesignReportInPanel(Panel1);
07  Panel1.Repaint;
08 end;
```

Após implementar o método SalvarModelo, faremos o código FastReportRefresh, que pode ser visto na **Listagem 7**.

Utilizamos o método Repaint no componente Panel (linha 03 e linha 07). Este método tem o objetivo de atualizar as informações visuais de um componente VCL. A seguir, para o componente FrxReport simplesmente removemos qualquer valor pré existente na propriedade Designer (linhas 04 e 05) e por último, o método DesignReportInPanel é chamado, passando o objeto Panel1 como parâmetro. Isso fará com que o mecanismo de Designer do Fast Report seja exibido no objeto container passado como parâmetro.

A seguir, o método CarregarModelo deverá ser implementado em sua aplicação conforme a **Listagem 8**.

Nota do DevMan 1

TypeCast é uma técnica muito utilizada em Delphi em conjunto com RTTI (RunTime Type Information). O objetivo é associar um determinado objeto a outro tipo que não seja sua classe de criação. Exemplo:

(Sender as TWinControl) ou TWinControl(Sender)

Neste caso o objeto Sender (que poderia ser um TObject) é dito como um tipo TWinControl, no entanto, este tipo de conversão não é totalmente seguro, podendo assim ocasionar erros de conversão inválida. Uma alternativa para lidar com a manipulação de objetos de forma segura é através de Generics.

Listagem 8. Método CarregarModelo

```
01 procedure TFrmPrincipal.CarregarModelo;
02 var
03  reportpage: TfrxComponent;
04 begin
05  cdsEtiquetas.Open;
06  reportpage := FrxReport.FindObject('Page1');
07  if (reportpage.ClassType = TfrxReportPage) then
08  begin
09   (reportpage as TfrxReportPage).PaperHeight :=
     cdsEtiquetasALTURA_PAG.AsVariant;
10   (reportpage as TfrxReportPage).PaperWidth :=
     cdsEtiquetasLARGURA_PAG.AsVariant;
11   (reportpage as TfrxReportPage).LeftMargin :=
     cdsEtiquetasMARGEM_ESQUERDA_PAG.AsVariant;
12   (reportpage as TfrxReportPage).RightMargin :=
     cdsEtiquetasMARGEM_SUP_PAG.AsVariant;
13   (reportpage as TfrxReportPage).TopMargin :=
     cdsEtiquetasMARGEM_SUP_PAG.AsVariant;
14   (reportpage as TfrxReportPage).BottomMargin :=
     cdsEtiquetasMARGEM_INF_PAG.AsVariant;
15  end;
16  reportpage := FrxReport.FindObject('MasterData1');
17  if (reportpage.ClassType = TfrxMasterData) then
18  begin
19   (reportpage as TfrxMasterData).Columns :=
     cdsEtiquetasNUM_COLUNAS.AsVariant;
20   (reportpage as TfrxMasterData).ColumnWidth :=
     cdsEtiquetasLARGURA_ETIQUETA.AsVariant * 37.8;
21   (reportpage as TfrxMasterData).Height :=
     cdsEtiquetasALTURA_ETIQUETA.AsVariant * 37.8;
22   (reportpage as TfrxMasterData).ColumnGap :=
     cdsEtiquetasDISTANCIA_HORIZONTAL.AsVariant * 37.8;
23  end;
24  reportpage := FrxReport.FindObject('Filha');
25  if (reportpage.ClassType = TfrxChild) then
26  begin
27   (reportpage as TfrxChild).Height :=
     cdsEtiquetasDISTANCIA_VERTICAL.AsVariant * 37.8;
28  end;
29  reportpage := frxReport.FindObject('frxDBClientes1CLI_NOME');
30  if (reportpage.ClassType = TfrxMemoView) then
31  begin
32   (reportpage as TfrxMemoView).Height :=
     cdsEtiquetasALTURA_CAMPO_NOME.AsVariant;
33   (reportpage as TfrxMemoView).Width :=
     cdsEtiquetasLARGURA_CAMPO_NOME.AsVariant;
34   (reportpage as TfrxMemoView).Top :=
     cdsEtiquetasTOP_CAMPO_NOME.AsVariant;
35   (reportpage as TfrxMemoView).Left  :=
     cdsEtiquetasLEFT_CAMPO_NOME.AsVariant;
36  end;
37  reportpage := FrxReport.FindObject('frxDBClientes1CLI_TELEFONE');
38  if (reportpage.ClassType = TfrxMemoView) then
39  begin
40   (reportpage as TfrxMemoView).Height :=
     cdsEtiquetasALTURA_CAMPO_FONE.AsVariant;
41   (reportpage as TfrxMemoView).Width :=
     cdsEtiquetasLARGURA_CAMPO_FONE.AsVariant;
42   (reportpage as TfrxMemoView).Top :=
     cdsEtiquetasTOP_CAMPO_FONE.AsVariant;
43   (reportpage as TfrxMemoView).Left :=
     cdsEtiquetasLEFT_CAMPO_FONE.AsVariant;
44  end;
45  FastReportRefresh;
46 end;
```

Ela define o processo inverso do que foi feito na **Listagem 7** (método SalvarModelo), sendo assim, não entraremos em grandes detalhes. Nela podemos perceber que o componente cdsEtiquetas é aberto (linha 05). A seguir, na linha 06, o processo de obter o ponteiro do objeto através do método FindObject é novamente utilizado, posteriormente são checados os tipos caso a caso e assim, os valores são passados para as propriedades do relatório. Ao final de todo o processo o método FastReportRefresh é novamente chamado (linha 45).

Até aqui com cinco clientes cadastrados no banco e a aplicação de etiquetas já criada, podemos visualizar como será a impressão. Para isso basta utilizar o botão Preview, conforme a **Figura 8.**

Aqui cabe um questionamento interessante onde você pode se perguntar: "como o cliente irá modificar o tamanho das margens, folha e todos aqueles campos que eu não adicionei no Edit?". Pois bem, aqui que entra em uso o editor do próprio Fast Report. Para isso, basta dar um clique de direita sobre a página e selecionar a opção editar (**Figura 9**).

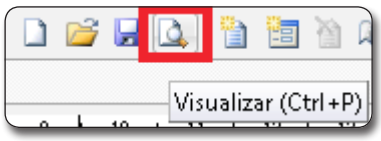

Figura 8. Botão preview

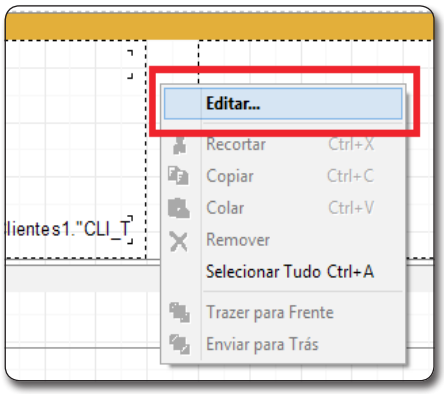

Figura 9. Editando o relatório

Finalizando a codificação de nossa aplicação, basta codificar os eventos OnCreate e OnShow do formulário (**Listagens 9** e **10** respectivamente).

Na **Listagem 9** podemos perceber que simplesmente os ClientDataSets cdsClientes e cdsEtiquetas são abertos (linhas 03 e 04) para a seguir, o método DesignReportInPanel ser executado, passando o painel como parâmetro.

O evento OnShow (**Listagem 10**) simplesmente dispara o método inicial CarregarModelo.

Ao final do artigo, o resultado final será o mesmo da **Figura 10**. Observe que para as barras laterais e menus do Fast Report desaparecerem basta ir ao menu View>Toolbars e desmarcar todas as opções.

Listagem 9. Evento OnCreate do formulário

```
01 procedure TFrmPrincipal.FormCreate(Sender: TObject);
02 begin
03  cdsClientes.Open;
04  cdsEtiquetas.Open;
05  FrxReport.DesignReportInPanel(Panel1);
06 end;
```

Listagem 10. Evento OnShow do formulário

```
01 procedure TFrmPrincipal.FormShow(Sender: TObject);
02 begin
03  CarregarModelo;
04 end;
```

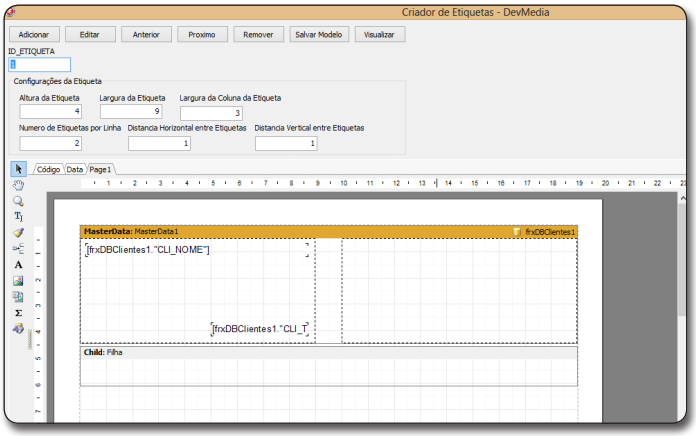

Figura 10. Aparência final do sistema

Conclusão

Este foi um pequeno exemplo de como podemos criar um editor de modelos de etiquetas. Todo este processo abordado no artigo pode ser realizado com a finalidade de otimizar o tempo e diminuir o suporte. Com o editor de relatórios dinâmico é possível personalizar as etiquetas conforme necessidade. Espero que tenham gostado. Um grande abraço a todos.

Rafael Isique Guardieri Brendolan
brendolan@outlook.com
Cursando Sistemas de Informação pela São Luis - Jaboticabal. Atua como analista e programador de sistemas da equipe de desenvolvimento Delphi da Siplan Control-M.

Dê seu feedback sobre esta edição!

A ClubeDelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/clubedelphi/feedback

Certificação Delphi – Parte 2

Conheça o novo programa de certificação da Embarcadero

Este artigo faz parte de um curso

Resumo DevMan

Certificação Delphi :

No novo programa elaborado pela Embarcadero, a certificação Delphi é segmentada em dois níveis de dificuldade, caracterizando assim dois tipos disponíveis: Delphi Developer e Delphi Master Developer. A primeira, tema central da parte um desta série de artigos, é considerada a mais básica e seu nível de exigência se dá pelos conceitos básicos e essenciais que envolvem a programação Delphi. Já nesta segunda e última parte da série é então apresentada a certificação Delphi Master Developer que, numa evolução natural, pode ser considerada a continuação do caminho trilhado até a Delphi Developer. Sendo assim, serão vistos os detalhes que envolvem sua realização, seus requisitos e, fundamentalmente, os tópicos de estudo relacionados. Novamente, ao final do artigo, serão apresentadas algumas questões de exemplo, aos moldes das presentes no exame.

Em que situação o tema é útil:

A busca por uma certificação em determinada linguagem ou tecnologia extrapola o lado profissional e financeiro, atingindo também o lado pessoal. De certa forma, colocar seus conhecimentos à prova pode significar a satisfação do próprio ego e até mesmo a realização de um sonho. Sendo assim, a Certificação Delphi pode ser considerada um mix entre destaque profissional e satisfação pessoal. Dentre as certificações disponíveis, a Delphi Master Developer se credencia como sendo a mais avançada, voltada ao desenvolvedor Delphi mais experiente. Os detalhes que envolvem sua conceituação e realização serão então vistos no presente artigo.

O novo programa de certificação da Embarcadero, lançado ainda na época da versão XE do Delphi, representou a volta da "Certificação Delphi". De forma prática, este programa estabeleceu no mercado duas novas nomeações relacionadas à certificação – Delphi Developer e Delphi Master Developer. Apesar de ambas estarem relacionadas ao mesmo contexto principal, o Delphi, cada uma apresenta suas próprias nuances, o que resulta em níveis de dificuldade bem distintos. Toda a parte técnica, principalmente no que diz respeito à linguagem e ao IDE, é fundamentada até a versão XE da ferramenta. Isso significa dizer que novas características e recursos introduzidos posteriormente, vide XE2 e XE3, não estão inclusas no programa.

Certificação – Delphi Master Developer

Ao contrário da certificação Delphi Developer, que testa e exige um nível de conhecimento mais básico/intermediário do candidato, a certificação Delphi Master Developer prega por conhecimentos mais avançados sobre os conceitos relacionados à programação Delphi. Dessa forma, aspectos sobre arquitetura de Software, técnicas de programação, desenvolvimento efetivo de aplicações (Desktop, Internet e Multicamadas), linguagem, IDE e componentes são exigidos.

Requisitos para a realização da prova

A certificação Delphi Master Developer é tida como de nível avançado, sendo assim, o tempo de experiência mínimo em programação Delphi sugerido pela Embarcadero para a realização do exame é de quatro anos. Com este tempo especulado de vivência com a ferramenta e a linguagem, o desenvolvedor muito provavelmente já extrapolou o nível básico de conhecimento no desenvolvimento Delphi em geral. Em complemento, espera-se que o candidato siga o guia de estudos provido pela própria Embarcadero, como forma de aprofundar os tópicos de conhecimentos exigidos.

Preparação

Como pontapé inicial para a fase de preparação do exame da certificação Master Developer, a Embarcadero disponibiliza em seu site (ver seção **Links**) um guia de estudos (Study Guide) contendo um roteiro de recursos que irão contribuir no aperfeiçoamento de seus conhecimentos. Dentre estes recursos estão indicações de livros de autores consagrados no meio Delphi, links para Websites especializados, links para tópicos específicos da documentação online do produto, bem como sugestões de treinamentos oficiais presenciais.

Em se tratando de treinamentos, estes são oferecidos pela própria Embarcadero por meio de seus parceiros espalhados pelo mundo. Só no Brasil são dezenas de opções. Dentre os cursos oferecidos, a grande maioria tem seu conteúdo condizente com os tópicos requeridos pela certificação Master Developer, uma vez que a mesma é bastante abrangente em termos de conceitos. Desenvolvimento de Aplicações Win32 (**Nota do DevMan 1**) e DataSnap Multitier (**Nota do DevMan 2**) são alguns exemplos destes treinamentos oficiais.

Sem deixar de mencionar, o aspecto do idioma também deve ser levado em conta durante a fase de preparação para a realização do exame, uma vez que a prova é toda em inglês.

Nota do DevMan 1

Desenvolvimento de Aplicações Win32 é o nome do treinamento oficial da Embarcadero voltado aos profissionais que já possuem conhecimento em Delphi e que queiram agregar novos conhecimentos ou mesmo se reciclar. Seu foco principal é o desenvolvimento de aplicações Win32 com banco de dados. Seu conteúdo programático inclui: IDE, OO, Regras de negócio, UML, Design Patterns, elementos de UI, tratamento de exceções, depuração, pacotes, dbExpress, TClientDataSet, gráficos, relatórios e desenvolvimento avançado com banco de dados.

Nota do DevMan 2

DataSnap Multitier é o nome do treinamento oficial da Embarcadero voltado aos desenvolvedores Delphi já habituados à construção de sistemas Desktop (Client/Server) e que desejam evoluir seus conhecimentos para o desenvolvimento multicamadas. Seu foco principal é o desenvolvimento de aplicações Delphi apoiados pela tecnologia DataSnap. Seu conteúdo programático inclui basicamente DataSnap clássico, novo DataSnap, dbExpress, REST, JSON, Callbacks e outros conceitos relacionados.

Custo da prova

De acordo com o site da Embarcadero, o custo do exame para a certificação Delphi Master Developer é de U$149 dólares, mas este valor poderá estar diluído em uma oferta de treinamento em um de seus diversos centros autorizados.

Realização da prova

As certificações inclusas no novo programa da Embarcadero apresentam alguns pontos em comum, relacionados principalmente a quesitos como tempo e questões. Sendo assim, tanto a certificação Delphi Developer quanto a Delphi Master Developer determinam como 60 minutos o tempo limite para a realização do exame. O corpo de cada exame é então composto por 60 questões de múltipla escolha, todas em inglês, e apresentando o mesmo peso para a soma da pontuação final.

Diferentemente da Delphi Developer, que provê a realização da prova de forma totalmente online a partir de qualquer computador com acesso à internet, a prova da certificação Delphi Master Developer é presencial, sendo aplicada em uma das várias Centrais de Certificação espalhadas pelo país e no mundo. O link para a página da Embarcadero contendo a lista completa de centros aptos para a aplicação da prova pode ser vista na seção **Links** ao final do artigo.

Ainda na Central de Certificação, no momento da realização da prova, a identificação do candidato é validada de forma a garantir, a quem possa interessar, sua real identidade. Imediatamente ao final do teste o candidato já toma conhecimento de sua nota.

Outra diferença notória e esperada entre as certificações está relacionada ao seu conteúdo programático. Dito isto, as questões presentes no exame da certificação Delphi Master Developer são fundamentas nos seguintes tópicos de conhecimento:

- Conceitos de Windows;
- Interface e Configuração do Delphi XE;
- VCL (Visual Component Library);
- RTL (Run-Time Library);
- Componentes (Conceitos Básicos, Design e Manipulação);
- Linguagem Delphi (Delphi Language);
- Programação Orientada a Objetos;
- Conceitos de Banco de Dados;
- Técnicas de Acesso a Dados;
- dbExpress;
- DataSnap;
- Bibliotecas (DLLs) e Pacotes (Packages);
- Programação para Internet;
- XML.

Dentre esses tópicos, é garantido que ao menos uma questão será extraída de cada um para compor o corpo final do exame.

A aprovação

Em relação à aprovação no exame, novamente ambas as certificações, Master Developer e Developer, fazem uso do mesmo critério, exigindo um percentual mínimo de 80% de acertos, ou 48 questões corretas. Uma vez aprovado, o desenvolvedor passa a ser identificado pela própria Embarcadero como um membro de elite da comunidade Delphi espalhada pelo globo, sob o título de Certified Delphi Master Developer. Adicionalmente, o candidato aprovado ainda recebe dois itens para uso pessoal/profissional, representativos de sua certificação: um documento de certificação devidamente chancelado pela Embarcadero, bem como uma logo digital de Desenvolvedor Delphi Master Certificado.

Em termos de validade, a certificação Master Developer, assim como a Developer, tem a validade de dois anos, a partir da data da realização do exame.

Tópicos de estudo

De forma realista, a lista de tópicos de estudo para a certificação Master Developer é bastante extensa, sendo assim, o ideal é abordar cada item de forma única e dedicar o tempo necessário para o bom entendimento ou mesmo a releitura dos detalhes envolvidos. Isto porque, devido à grande quantidade de assuntos relacionados, muitos dos tópicos acabam por envolver conceitos não tão usuais no cotidiano do candidato, merecendo uma atenção especial na hora do estudo. Com base no Study Guide disponibilizado pela Embarcadero, a seguir é feita uma explanação sobre cada tópico de estudo listado anteriormente:

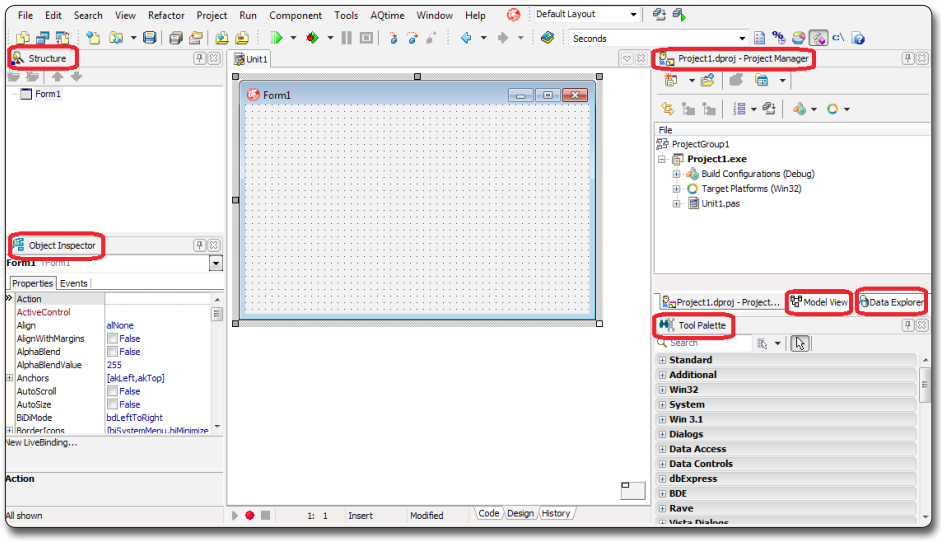

Figura 1. Visão geral do IDE do Delphi

Conceitos de Windows

Este tópico faz referência às tratativas relacionadas ao Windows em meio à programação Delphi. Sendo assim, o teste sobre seus conhecimentos irá passar por pontos como conhecimento geral da API, constantes base Windows, programação GDI, Handles e manipulação de mensagens do Windows. Além disso, uma prática um tanto quanto comum no desenvolvimento Delphi, que é o trabalho com o registro do Windows (Windows Registry), também é outro item incluso neste tópico. O trecho de código da **Listagem 1**, utilizado para mostrar/ocultar a barra de tarefas do Sistema Operacional, exibe um exemplo simples de uso dos pontos citados até aqui, tais como constantes Windows (Ex: SW_RESTORE), e métodos da API (Ex: ShowWindow), disponíveis no Delphi através da unit Windows.

Listagem 1. Método ShowHideToolBar

```
procedure TForm1.ShowHideToolBar(Show: Boolean);
var
  wndHandle : THandle;
  wndClass : array[0..10] of Char;
begin
  StrPCopy(@wndClass[0],'Shell_TrayWnd');
  wndHandle := FindWindow(@wndClass[0], nil);

  if Show then
    ShowWindow(wndHandle, SW_RESTORE) // Mostra a barra de tarefas do Windows
  else
    ShowWindow(wndHandle, SW_HIDE); // Oculta a barra de tarefas do Windows
end;
```

Interface e Configuração do Delphi XE

No desenvolvimento de Software, o termo "IDE" faz referência ao termo em inglês Integrated Development Environment que, numa tradução livre para o português significa Ambiente de Desenvolvimento Integrado. Numa definição rápida, podemos dizer que se trata de um Software composto de vários recursos integrados. Isso se encaixa perfeitamente ao Delphi, uma vez que seu ambiente é composto por diversas características e ferramentas distintas que irão apoiar o processo de desenvolvimento, proporcionando uma maior produtividade aos desenvolvedores.

Conforme o nome do tópico sugere, Interface e Configuração do Delphi XE diz respeito então ao conhecimento sobre este ambiente de desenvolvimento do Delphi. Em termos gerais, o candidato deverá possuir o devido conhecimento sobre cada janela de recurso presente no ambiente, tais como Code Editor, Project Manager, Model View, Object Inspector, Form Designer, Structure Window, Class Explorer, Data Explorer e Tool Palette. Além disso, outros pontos relacionados também são exigidos como, por exemplo, a personalização do IDE, configuração do depurador (Debugger) integrado, uso de recursos como Live Templates, Build Configurations, Option Sets, Project Groups, auditorias (Audits) e métricas (Metrics). Na **Figura 1** é mostrada uma visão geral do IDE do Delphi, onde podem ser vistas algumas das janelas de recursos mencionadas anteriormente.

VCL

VCL é o acrônimo para Visual Component Library, que remete à biblioteca de componentes visuais do Delphi. Ela é composta por uma grande variedade de classes, tanto visuais quanto não visuais, que contribuem de uma forma geral ao desenvolvimento rápido de aplicações com Delphi. De forma anexa à VCL, estão os componentes, que representam um subconjunto da biblioteca e descendem de TComponent. Sua característica mais marcante é sua manipulação em tempo

de design, provendo, entre outras coisas, o já tradicional comportamento de arrastar-e-soltar elementos nos formulários e Data Modules do projeto, além da atribuição de valores às suas propriedades através do Object Inspector, sem a necessidade de se escrever código algum. Um componente pode ainda ser classificado como visual (Ex: TButton) ou não-visual (Ex: TDataSource), variável à sua visibilidade em tempo de execução.

Aqui, os pontos de conhecimento exigidos do candidato englobam desde a hierarquia de classes da VCL, passando pela construção de aplicações VCL (as tradicionais VCL Forms Applications), chegando a componentes (visuais e não visuais) e suas propriedades, métodos e eventos relacionados.

RTL

Run-Time Library, ou simplesmente RTL, é a biblioteca de tempo de execução do Delphi. Sua estrutura é bem abrangente, uma vez que é composta de Units consideradas como base para outras bibliotecas, tais como a VCL e a recente FMX (FireMonkey). Apesar desta ligação, é importante ressaltar que a RTL não inclui qualquer componente disponível para uso no Delphi, ou seja, aqueles presentes na Tool Palette do IDE. No entanto, estes componentes é que fazem uso dos recursos providos pela RTL. De forma sucinta, podemos dizer que a RTL engloba rotinas e classes globais, que remetem ao topo de uma hierarquia, tais como TObject, TPersistent e TComponent (**Figura 2**).

Em termos gerais, o conhecimento sobre os detalhes das classes e rotinas providas pela RTL é o exigido neste tópico.

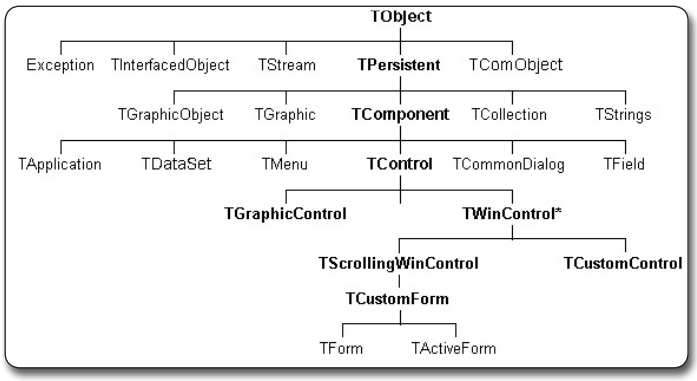

Figura 2. Visão geral da hierarquia de classes do Delphi

Componentes (Conceitos Básicos, Design e Manipulação)

Este tema pode ser considerado um complemento ao tópico VCL, uma vez que também trata de componentes, mas agora de uma forma mais específica. Os pontos abordados circundam a criação, utilização e Design de componentes. Tudo isso se reflete num primeiro momento, no conhecimento sobre propriedades básicas, métodos e eventos de componentes, sua utilização em projetos, interação entre os componentes da VCL, Actions e ActionLists. Além disso, outro conceito condizente a este cenário é o de Component Templates, que permite a criação de modelos de componente feitos a partir de um ou mais componentes. Tal recurso (**Figura 3**) fica disponível no menu Component>Create Component Template do IDE.

De forma complementar, especificamente relacionado ao termo Design, abre-se um novo leque de conceitos que poderão ser avaliados, tal como os seguintes:

- Criação de novos componentes;
- Extensão de componentes existentes;
- Adição e registro de novos componentes ao ambiente Delphi;
- Métodos de propriedade (Get e Set);
- Valores padrão de propriedades;
- Classes TComponent e TWinControl;
- Notificações de métodos;
- Component Editor e Property Editor;
- Expor/esconder propriedades e métodos de componentes.

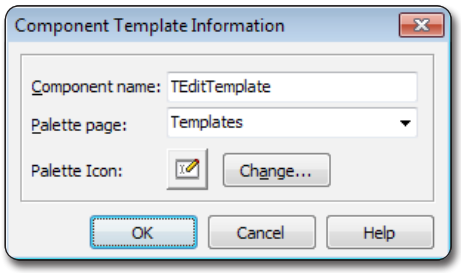

Figura 3. Recurso de Component Template

Linguagem Delphi

A Delphi Language é uma linguagem de programação baseada em Object Pascal e considerada de alto nível. É tida também como compilada e de fácil leitura, provendo recursos suficientes que dão todo o suporte necessário para projetos de Software que se utilizem tanto do paradigma estruturado quanto orientado a objetos. O uso da linguagem Delphi é sempre associado ao ambiente de desenvolvimento (IDE) Delphi.

Dito isto, este tópico irá testar os conhecimentos do candidato sobre as nuances da Delphi Language, tais como sua sintaxe, palavras reservadas, tipos de dados providos e suportados e convenções de chamada. Inclui-se ai o uso de algumas diretivas específicas da linguagem – deprecated, reintroduce, experimental e platform.

Programação Orientada a Objetos (POO)

De forma resumida, o conceito de Orientação a Objetos no desenvolvimento de Softwares pode ser definido como a busca pela aproximação do mundo real com o mundo da programação. Na POO, uma classe é a representação de um conjunto de objetos semelhantes enquanto que um objeto é simplesmente a instância de uma classe. No aspecto prático, a implementação de um conjunto de classes irá definir os objetos presentes na composição do Software.

Conforme dito, a linguagem Delphi provê o devido suporte à programação Orientada a Objetos e, por esta razão, este tópico acaba por complementar o anterior. Sendo assim, dentro do contexto Delphi, será exigido do candidato o devido conhecimento sobre

alguns dos conceitos da POO, tais como herança, polimorfismo, abstração, encapsulamento, sobrecarga de métodos, sobrecarga de operador, aninhamento de tipos, construtores (Constructors), destruidores (Destructors), escopo de classe e visibilidade de membros (**Listagem 2**).

Listagem 2. Visibilidades de membros para uma classe Delphi

```
TMyClass = class
  private
    { private declarations }
  protected
    { protected declarations }
  public
    { public declarations }
  published
    { published declarations }
  strict private
    { strict private declarations }
  strict protected
    { strict protected declarations }
  automated
    { automated declarations }
end;
```

Conceitos de Banco de Dados / Técnicas de Acesso a Dados

Na programação, um banco de dados pode ser definido com um conjunto de informações, capazes de se inter-relacionar e, assim, determinar um contexto. Por conseguinte, toda a gerência dos eventuais processos relacionados a um banco de dados é realizado por um Sistema de Gerenciamento de Banco de Dados, ou simplesmente SGBD. Dentro os inúmeros SGBDs disponíveis atualmente no mercado, podemos destacar Firebird, InterBase, Oracle, MySQL e SQL Server, todos suportados nativamente pelo Delphi.

No desenvolvimento Delphi propriamente dito, este suporte é então atrelado a recursos de acesso a dados. Sendo assim, ambos os tópicos, conceitos de banco de dados e técnicas de acesso a dados, acabam por compor um único. Aqui a exigência de conhecimento se faz pela criação de aplicações de banco de dados (Database Applications) com Delphi, classes TDataSet e TField, componentes de acesso a dados (relações e métodos), SQL, componentes Data-Aware, uso de transações, trabalho com Metadados, componentes IBX (InterBase Express) e componentes para ADO (grupo dbGo na Tool Palette – **Figura 4**).

dbExpress

dbExpress ou DBX é o nome de um dos principais frameworks incorporados ao Delphi cuja finalidade principal é prover às aplicações toda a parte de comunicação e acessibilidade a bancos de dados e Servidores de Aplicação (DataSnap). Por este fato, para o exame da certificação Master Developer, os conceitos exigidos relacionados à dbExpress se mostram bastante abrangentes, conforme se segue:

- Uso do framework dbExpress em aplicações Delphi;
- Arquivos dbxdrivers.ini e dbxconnections.ini;
- Debug de aplicações dbExpress;
- Deploy de aplicações dbExpress;
- Drivers e Delegates;
- Conexões e Pool de Conexões (Connection Pooling);
- Tratamento de exceções de banco de dados;
- Trabalho com Metadados;
- Trabalho com Transações;
- Uso dos componentes dbExpress;
- Relações de uso entre componentes dbExpress;
- Datasets unidirecionais;
- Utilização e configuração do componente TSQLConnection;
- Uso e conhecimento das propriedades, métodos e eventos dos componentes TSQLDataSet, TSQLQuery, TSQLTable e TSQLStoredProc.

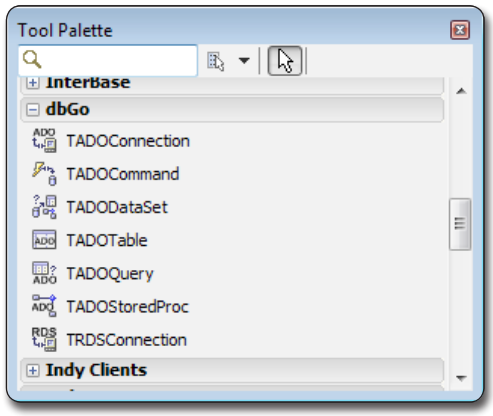

Figura 4. Grupo de componentes dbGo na Tool Palette

DataSnap

DataSnap (anteriormente tratado como MIDAS), é a tecnologia presente no Delphi voltada ao desenvolvimento de aplicações multicamadas (multi-tier). Neste cenário, há uma clara e notória divisão, tanto lógica quanto física, entre as partes que compõem o Software, tais como Dados, Regras de Negócio (**Figura 5**) e Cliente. Em razão disso, a característica predominante do DataSnap é a sua capacidade em tornar possível a chamada de um método remoto, localizado num Servidor de Aplicação, a partir de um determinado aplicativo cliente, de forma totalmente transparente e simplificada.

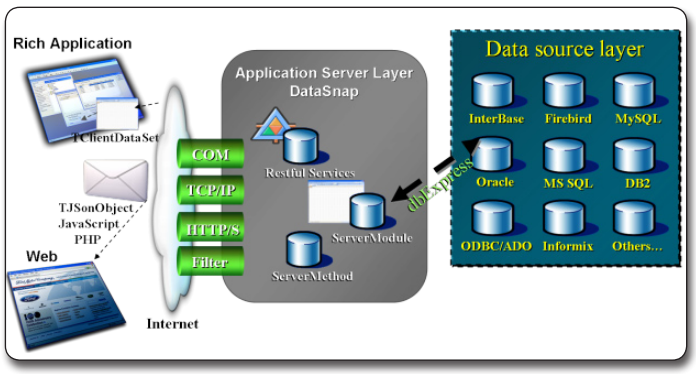

Figura 5. Exemplo clássico de uma arquitetura DataSnap

Isto posto, este tópico abrange de uma forma geral o desenvolvimento multicamadas no Delphi, com o uso da tecnologia DataSnap, conforme os seguintes pontos:
- Arquitetura DataSnap;
- Protocolos suportados;
- Gerenciamento de memória DataSnap;
- Lifecycle;
- Server Methods;
- Deploy de servidores DataSnap;
- Debug de servidores DataSnap;
- Callbacks;
- Filters;
- JSON;
- REST;
- Pool de Conexões (Connection Pooling);

Bibliotecas (DLLs) e Pacotes (Packages)

Uma DLL (Dynamic Link Library) nada mais é do que uma biblioteca que contém código e dados, responsáveis por prover funcionalidades ao aplicativo ao qual estiver trabalhando em conjunto. De forma similar, um pacote (Package) pode ser considerado uma DLL especial voltada exclusivamente ao contexto Delphi, sendo então utilizada tanto por aplicações Delphi ou mesmo pelo próprio IDE.

Aqui, o foco se dá sobre a escrita de DLLs em Delphi, bem com a utilização efetiva de pacotes. Em vista disso, os itens requeridos para DLL giram em torno de:
- Escrita de DLLs em Delphi;
- Tipos de uso;
- Dados globais (global data);
- Diretiva Delayed;
- Deploy;

Da mesma forma, os itens relacionados a Packages são:
- Tipos de pacotes (Runtime Packages e Design-Time Packages);
- Estrutura de pacotes;
- Convenções de nomenclatura para pacotes;
- Cláusulas Contains e Required;
- Referências circulares;
- Deploy.

Programação para Internet

Este tópico abrange um assunto um tanto quanto desconhecido no meio Delphi, que é sua capacidade de programação de aplicativos voltados à internet. Isto se justifica pelo fato do grande chamariz do Delphi estar centrado em sua excelência na produção de aplicativos Desktop. Todavia, é importante ressaltar que a ferramenta provê recursos voltados à programação para Internet desde suas versões iniciais. Prova disso são os componentes Web Broker, localizados na guia Internet da Tool Palette do IDE (**Figura 6**) e comumente utilizados para a construção de aplicativos de Servidor Web (Web Server Applications).

Mediante isso, conceitos fundamentais relacionados à Web, tais como HTML, CSS, URL, URI, JSON, tipos de solicitações HTTP e protocolos (TCP, HTTP e HTTPS) são obviamente exigidos. Somam-se a eles alguns outros conceitos, agora condizentes ao contexto Delphi:
- Consumo de WSDL (Web Services);
- Aplicações CGI e ISAPI;
- Web Services SOAP e RESTFul;
- Web Server Applications;
- Componentes relacionados à Internet providos pelo Delphi XE;
- Web Modules;
- Web Broker;
- Web Actions;
- Web Dispatcher.

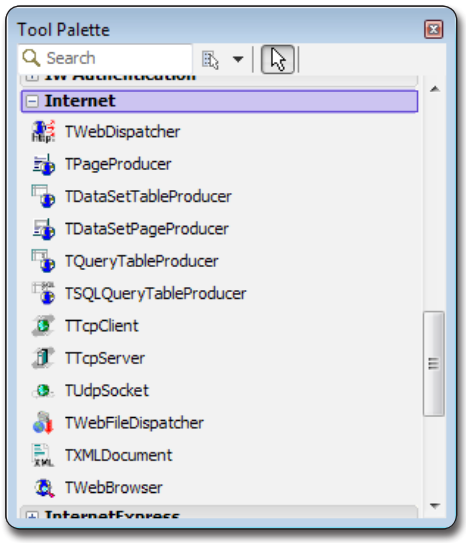

Figura 6. Componentes Web Broker na guia Internet da Tool Palette

XML

XML (eXtensible Markup Language) pode ser definida como sendo uma linguagem de marcação padrão e extensível, utilizada na criação de documentos cujo conteúdo é definido com dados organizados de forma hierárquica. Por se tratar de um elemento muito utilizado no meio computacional, ele é plenamente suportado pelo Delphi, sendo utilizado para os mais variados fins. Como exemplo, o código da **Listagem 3** exibe o código de um método que faz uso das classes relacionadas à XML providas pelo Delphi para a criação de um arquivo XML. O resultado proporcionado por esta listagem, ou seja, o conteúdo do arquivo gerado por ela pode ser visto na **Figura 7**.

Para a certificação Master Developer, os conceitos exigidos relacionados ao emprego de XML no cenário Delphi são:
- Sintaxe XML;
- Documentos relacionados à XML - XML Document Object Model (DOM), Document Type Definition (DTD), XML Schema Definition (XSD) e XML Data Reduced (XDR);
- Aplicações CGI e ISAPI;

• Componentes e classes utilizadas em aplicações baseadas em XML;
• XML Transforms;
• Validação de documentos XML;
• XML Nodes;
• XML Parsing;
• XML Data Binding;
• XML e Data Packet;
• XML e ClientDataSet.

Listagem 3. Código de um método que gera um arquivo XML

```
01 procedure TForm1.GeraXML;
02 var
03  XMLDoc: IXMLDocument;
04  XMLNode, IdentificacaoNode, ConteudoNode: IXMLNode;
05 begin
06  // Cria um XML Document
07  XMLDoc := NewXMLDocument;
08  XMLDoc.Encoding := 'ISO-8859-1';

09  // Adiciona um XML Node ao documento
10  XMLNode := XMLDoc.AddChild('REVISTA');

11  // Adiciona um XML Node dentro do Node REVISTA
12  IdentificacaoNode := XMLNode.AddChild('IDENTIFICACAO');

13  // Adiciona dois XML Node dentro do Node IDENTIFICACAO
14  ConteudoNode := IdentificacaoNode.AddChild('NOME');
15  ConteudoNode.Text := 'CLUBE DELPHI';
16  ConteudoNode := IdentificacaoNode.AddChild('ANO');
17  ConteudoNode.Text := '2013';

18 // Salva arquivo XML em disco
19  XMLDoc.SaveToFile('C:\DevMedia\Teste.xml');
20 end;
```

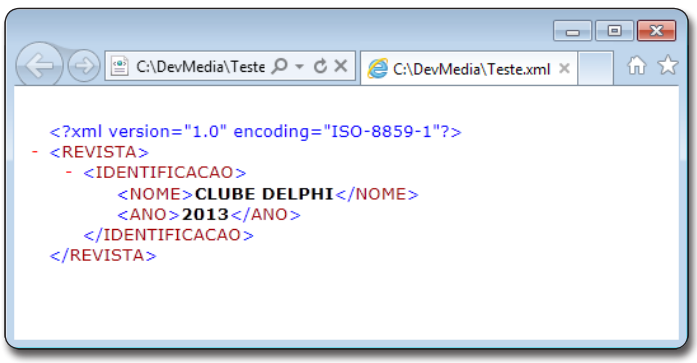

Figura 7. Conteúdo do arquivo XML gerado pelo método

Perguntas e Respostas

A seguir são expostas, de forma traduzida, algumas perguntas/respostas extraídas do Study Guide provido pela Embarcadero para a certificação Delphi Master Developer. Para cada questão é tecido um breve comentário contendo a justificativa para a resposta assinalada.

1. Você precisa criar um novo componente que é capaz de receber o foco e interagir com o teclado. Qual das seguintes classes é a melhor ancestral para ser usada?

a) TControl
b) TWinControl
c) TGraphicControl
d) TComponent

Resposta: B.

Justificativa: Segundo o próprio Help do Delphi, a classe TWinControl é a responsável por prover funcionalidades comuns a todos os controles que atuam como objetos em janelas "Windows". Por esta razão, esses controles podem apresentar algumas características consideradas "padrão", tal como a capacidade de receber o foco e lidar com eventos de entrada de usuário.

2. Suponha que você execute um procedimento armazenado (Stored Procedure) usando o componente TSQLStoredProc, e ele retorne vários conjuntos de resultados (result sets). Como você navega para o próximo result set?

a) Executando o método NextRecordSet do componente
b) Executando o método Next do componente
c) TSQLStoredProc não suporta navegação sobre result sets
d) Nenhuma das anteriores

Resposta: A.

Justificativa: Uma Stored Procedure pode retornar um ou vários conjuntos de registros/resultados (result sets). Para este último, o componente TSQLStoredProc provê o método NextRecordSet, que permite se acessar sempre o próximo conjunto de registros a ser retornado. Sendo assim, NextRecordSet retorna um objeto do tipo TCustomSQLDataSet, que irá representar então o próprio conjunto de registros. Na prática, ao chamar NextRecordSet pela primeira vez, o segundo result set é retornado. Numa segunda chamada, o terceiro result set é retornado, e assim por diante.

3. Quais dos seguintes tipos são válidos em um método SOAP do lado servidor (Escolha todas que se aplicam)?

a) Integer
b) TDateTime
c) TStringList
d) Pointer

Resposta: A e B.

Justificativa: A arquitetura das SOAP Applications no Delphi provê suporte a tipos como Boolean, Integer, Char, String, TDateTime, Byte, Word, Double, Variant, entre outros. TStringList e Pointer não são suportados.

Conclusão

Do ponto de vista mais simbólico, a conquista de uma certificação pode significar a satisfação pessoal por um objetivo, ou mesmo a realização de um sonho maior. Não obstante, do ponto de vista profissional, tal conquista pode se tornar o trunfo que

faltava para se alcançar um status de destaque em meio a uma comunidade. Por estes motivos, a dedicação de tempo e esforços em busca desta conquista deve ser vista sempre com bons olhos pelo desenvolvedor.

Nesta série de artigos, vimos então os detalhes que envolvem o novo programa da Embarcadero de certificações Delphi. No estágio atual, são duas as opções disponíveis aos desenvolvedores e amantes desta tecnologia: Delphi Developer, voltado ao nível intermediário, e a Delphi Master Developer, para o público mais avançado.

Agradeço a todos e até a próxima.

Fabrício Hissao Kawata
fabricio.kawata@bol.com.br
Formado em Processamento de Dados pela FATEC-TQ. Atua como Analista Programador Delphi há 8 anos. Atualmente escreve artigos sobre Delphi para revistas especializadas. Certified Delphi Developer. Certified Embarcadero Instructor. Blog: prismsilver.blogspot.com

Delphi XE3 – Download Trial
https://downloads.embarcadero.com/free/delphi

Delphi XE3 – Página oficial do produto
http://www.embarcadero.com/products/delphi

Embarcadero Delphi Master Developer Certification
http://www.embarcadero.com/br/certification/delphi-master

Embarcadero Certification Code Activation
http://certification.embarcadero.com/reg

Atendimento - Embarcadero do Brasil
http://www.embarcadero.com/br/how-to-buy

Embarcadero - Centrais de Certificação
http://www.embarcadero.com/br/company/contact-sales-find-partner?filtertype=Certification%20Center

Dê seu feedback sobre esta edição!

A ClubeDelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/clubedelphi/feedback

Conhecimento faz diferença!

Faça já sua assinatura digital ! | www.devmedia.com.br/es

Faça um *upgrade* em sua carreira

Em um mercado cada vez mais focado em qualidade, ter conhecimentos aprofundados sobre requisitos, metodologia, análises, testes, entre outros, pode ser a diferença entre conquistar ou não uma boa posição profissional. Sabendo disso a DevMedia lançou uma publicação totalmente especializada em Engenharia de Software. Todos os meses você pode encontrar artigos sobre Metodologias Ágeis; Metodologias tradicionais (document driven); ALM (application lifecycle); SOA (aplicações orientadas a serviços); Análise de sistemas; Modelagem; Métricas; Orientação à Objetos; UML; testes e muito mais. **Assine Já!**

DevMedia

RIA com Delphi – Parte 2

Aprenda criar aplicações ricas sem conhecer HTML

Este artigo faz parte de um curso

Nos artigos anteriores sobre RIA (Rich Internet Application – Aplicações Ricas para a Internet), foram a s empregadas para o desenvolvimento de Softwares interativos cuja plataforma alvo é a Web, tais como as linguagens JavaScript e HTML, os pilares para este tipo de aplicação. Também foram demonstrados alguns dos principais recursos do framework uniGUI, uma importante ferramenta para a elaboração de aplicações RIA através do Delphi, composto por um conjunto de componentes visuais e não visuais que aumentam consideravelmente a produtividade. Lembrando que o mesmo é gratuito, de fácil instalação e que seu motor Web é baseado no sólido e popular framework JavaScript, ExtJS, onde o ExtPascal é utilizado para unir ambas as tecnologias, como forma de um gateway entre o Delphi e o ExtJS, além do uniGUI gerar automaticamente toda a codificação front-end da aplicação RIA. Vale à pena evidenciar também que no artigo anterior foi apresentado um Software baseado neste framework que emprega os recursos da Google Maps API, possibilitando a exibição e manipulação de mapas na própria aplicação RIA, o que demonstra a flexibilidade do uniGUI, uma vez que além de gerar a codificação HTML e JavaScript, ele ainda permite a adição e integração de códigos desenvolvidos manualmente ou importados de terceiros, de uma forma rápida e simplificada através das propriedades de seus componentes, sendo esta uma funcionalidade extremamente importante, pois, demonstra que não são impostos limites ao desenvolvimento dos projetos. Contudo, até este momento, foram apresentados apenas exemplos de aplicações autônomas (standalone), que possuem algumas características próprias, tal como o fato de serem compostas pelo servidor Web e aplicação RIA em um único executável. Vale lembrar que dependendo do caso este é um fator que pode não ser o mais indicado para este tipo de aplicação, visto que no ambiente de Internet é possível notar que existe uma grande popularidade de servidores para a hospedagem de sites e aplicações Web que, contam com recursos mais avançados e diversas vantagens com relação a uma

Resumo DevMan

RIA com Delphi - Parte 2:

De fato, a Internet se tornou algo indispensável no cotidiano de muitas pessoas e organizações. Está presente nos computadores, dispositivos móveis, televisores, videogames, e assim por diante. É evidente que ela é um ambiente que oferece grandes possibilidades e, consequentemente, uma plataforma que vale a pena investir quando o assunto é desenvolvimento de aplicações. Afinal, o desenvolvedor elabora o Software apenas uma vez e, implementando todas as padronizações que existem neste ambiente e os recursos de uma arquitetura multicamadas, consegue que o mesmo seja acessado e consumido por qualquer dispositivo que possua um navegador e esteja conectado a rede, além de não existir a necessidade de percorrer máquina por máquina para possibilitar que as mesmas executem a aplicação (diferente da forma que ocorre com os tradicionais Softwares Desktop destinados a outros ambientes), onde cada dispositivo precisa receber configurações e instalações extras de acordo com as necessidades do projeto. Como já mencionado anteriormente, o desenvolvedor habituado com o Delphi que deseja aderir ao desenvolvimento para Web, não necessita sequer migrar de ferramenta ou ter grande domínio sobre as linguagens envolvidas, HTML e JavaScript, bastando apenas efetuar a instalação de alguns excelentes frameworks da atualidade para esta tarefa, como é o caso do uniGUI e do Raudus. Neste artigo é apresentada uma visão geral sobre a tecnologia ISAPI, bem como uma abordagem sobre a instalação e configuração do servidor Web Apache, para posteriormente demonstrar um exemplo de aplicação RIA com o uniGUI baseada em ISAPI para ser executada através deste servidor, facilitando o processo de disponibilização da aplicação através da Internet. Também é apresentada uma visão geral sobre o framework Raudus, demonstrando como elaborar uma aplicação integrada a um banco de dados para o ambiente da Internet, empregando recursos que vão um pouco mais além, tais como o download de determinadas informações e a impressão de relatórios, abrindo as possibilidades para que o desenvolvedor crie uma aplicação RIA com foco comercial.

Em que situação o tema é útil:

Possibilitar o desenvolvimento de aplicações ricas para a Internet e Intranet de uma forma simples e rápida através dos frameworks uniGUI e Raudus, usufruindo dos recursos que compõem o Delphi para esta tarefa e obtendo, desta forma, um Software compatível com os principais servidores Web da atualidade, através da criação de um módulo, facilitando o acesso, a manutenção, a migração, dentre outros recursos para a aplicação.

aplicação autônoma, uma vez que eles são direcionados para esta tarefa. Algumas das vantagens que podemos citar é a adição de módulos, segurança, personalizações, entre outros, como é o caso do Apache, que é gratuito, do IIS (Internet Information Services) que pertence a Microsoft, dentre outros. Sendo assim, a elaboração de aplicações que são compatíveis com estes Softwares servidores acaba se tornando a forma mais indicada para o desenvolvimento dentro deste contexto.

Como mencionado anteriormente, além de permitir a criação de uma aplicação RIA e Desktop autônoma, o uniGUI também permite a elaboração de módulos ISAPI (Internet Server Application Programming Interface). Em suma, um módulo ISAPI pode ser definido como um Software que é carregado e disponibilizado por um servidor HTTP. No caso do uniGUI, vale à pena lembrar que ele necessita das bibliotecas do ExtJS, que são adicionadas automaticamente no diretório do framework durante a instalação do mesmo.

Diante destes fatores, o desenvolvedor pode contar com várias alternativas para implementar aquela que mais se adéqua a sua necessidade, sendo este um diferencial muito importante do uniGUI, assim como do Raudus. Este último é um excelente grupo de ferramentas que também permite usufruir de todas as vantagens do ambiente RAD do Delphi para esta finalidade, gerando um servidor de aplicação autônomo ou um módulo para o servidor Apache, onde a aplicação cliente pode executar facilmente em vários navegadores presentes nos computadores (Cross browser) sem adição de quaisquer plug-ins, contando com funcionalidades e temas visuais que permitem uma aparência mais amigável quando a mesma for utilizada em plataformas móveis, através de um dispositivo baseado no Sistema Operacional Android ou iOS, por exemplo. Este framework também está disponível para o Lazarus.

ISAPI

O ISAPI é um recurso presente no ambiente Windows que possibilita o carregamento de funcionalidades adicionais como uma extensão, que pode ser uma aplicação RIA, ou um filtro para um servidor Web, permitindo assim aumentar ou aperfeiçoar as funcionalidades do mesmo. Para usufruir desta tecnologia, pode-se criar um módulo através de uma ferramenta de desenvolvimento, como é o caso do Delphi e utilizá-lo em conjunto com um servidor HTTP que ofereça suporte a este recurso, como é caso do Apache, do IIS, dentre outros. Um módulo ISAPI, como mencionado anteriormente, é na verdade uma biblioteca que contém determinado Software, neste contexto a aplicação Web RIA ou a maior parte dela. Em outras palavras, é um arquivo do tipo DLL (Dynamic Link Library), que por sua vez é carregado em um determinado processo gerenciado pelo servidor Web.

O acesso da aplicação RIA baseada em ISAPI por um cliente (navegador web) também é realizado através de uma URL, tal como o endereço de exemplo:

```
"http://localhost/Projeto/Aplicacao.dll/<Parâmetros>"
```

Onde simplesmente é informado o caminho para o módulo (arquivo DLL) que pode ou não conter parâmetros.

Vale à pena mencionar que aplicações baseadas em módulos ISAPI também podem acessar informações em bancos de dados normalmente, assim como ocorre com outros Softwares, evidentemente mediante a implementação dos recursos necessários.

Preparando o Apache para a aplicação RIA uniGUI

O Apache é um servidor HTTP multiplataforma gratuito, amplamente utilizado na Internet para a hospedagem dos sites e aplicações, que pode ser facilmente obtido através da página da Apache Software Foundation (ver seção **Links**). Dentre as características do projeto, estão a segurança, o desempenho, a eficiência e a flexibilidade, fatores de extrema importância para um servidor Web. Um ponto importante sobre o Apache é com relação ao seu funcionamento baseado em módulos, onde é possível definir o que será ou não carregado pelo servidor, bem como adicionar funcionalidades e estender seus recursos de uma forma simples e rápida, como é o caso de empregar suporte a linguagem de programação server side PHP (PHP: Hypertext Preprocessor).

Uma vez que o Apache tenha sido obtido, basta efetuar sua instalação seguindo o processo padrão "Next-Next-Finish". Durante este momento são solicitadas algumas informações, tais como domínio, e-mail, entre outras, porém, são dados que podem ser facilmente alterados posteriormente. Concluída a instalação, é criado um novo processo, responsável pelo servidor Apache, o httpd.exe, que pode ser facilmente monitorado ou parado pelo gerenciador de tarefas do Windows (CTRL + ALT + DEL), assim como seu serviço pode ser facilmente iniciado através de "Painel de Controle>Ferramentas Administrativas>Serviços" ou dos atalhos que constituem o grupo de programas Apache, presente no Menu Iniciar.

O próximo passo para dar continuidade ao projeto é efetuar a configuração do servidor HTTP. No Apache, a configuração é realizada através da edição de um arquivo baseado em texto, denominado httpd.conf, presente no diretório "C:\Program Files (x86)\Apache Software Foundation\Apache2.2\conf", que também pode ser facilmente acessado através do Menu Iniciar. É importante frisar que todas as vezes que este arquivo for editado, para que as alterações possam surtir efeito, o serviço do apache deve ser reiniciado para que as configurações sejam devidamente carregadas. Com o arquivo httpd.conf aberto, deve-se primeiramente definir o diretório onde se encontram os projetos, em outras palavras, as páginas de Internet ou as aplicações Web para que as mesmas possam ser disponibilizadas pelo servidor. Sendo assim, basta procurar pela linha iniciada com "DocumentRoot" e atribuir o endereço desejado, tal como o trecho a seguir:

```
DocumentRoot "C:/Sites"
```

Como o framework uniGUI possibilita a criação de módulos ISAPI, o servidor Apache também deve ser configurado para oferecer suporte a esta tecnologia. O primeiro passo para habilitar

este recurso, ainda no arquivo httpd.conf, é remover o caractere "#" do início da linha exibida na sequência:

```
LoadModule isapi_module modules/mod_isapi.so
```

Ao remover o comentário dessa linha, o apache efetuará o carregamento do módulo "mod_isapi.so", responsável pelo suporte aos recursos ISAPI, salvo que em algumas versões ela já pode estar descomentada. A próxima etapa é adicionar outras configurações para o correto funcionamento da aplicação RIA. A **Listagem 1** apresenta as configurações adicionadas ao arquivo de configuração do Apache.

Listagem 1. Configurações adicionais para o httpd.conf

```
AddHandler isapi-handler .dll

<Directory "C:/Sites">
  Options Indexes FollowSymLinks ExecCGI
  AllowOverride None
  Order allow,deny
  Allow from all
</Directory>

Alias /aplicacoes "C:/Sites"
```

No início deste trecho, através de AddHandler é efetuada a associação de arquivos do tipo DLL com o manipulador ISAPI, isapi-handler. Posteriormente, através de <Directory>, o diretório que contém os módulos, neste exemplo "C:/Sites", é adicionado à entrada de diretórios do Apache, contendo algumas configurações para que a aplicação RIA baseada em ISAPI possa ser executada. Para concluir, é criado um Alias denominado "aplicacoes" apontando para o diretório "C:/Sites", lembrando que um Alias pode ser resumido como um "apelido" para identificar determinado recurso, tal como a representação de um endereço, uma interface de rede, etc.

Outra configuração interessante que pode ser adicionada ao arquivo httpd.conf é a ISAPICacheFile, conforme o trecho a seguir:

```
ISAPICacheFile C:/Sites/<aplicação.dll>
```

Esta configuração possibilita que um módulo ISAPI possa ser pré-carregado e mantido na memória, uma alternativa que pode ser utilizada quando uma aplicação RIA tende a ser acessada várias vezes.

Para finalizar esta etapa, o passo seguinte é copiar a pasta que contém os arquivos do framework ExtJS, ext-3.4.0, a base para o funcionamento dos recursos do uniGUI, presente no diretório "C:\Program Files (x86)\FMSoft\Framework\uniGUI\", para a pasta onde o módulo ISAPI será armazenado, neste caso o diretório "C:\Sites", conforme definido no arquivo de configuração do Apache.

Terminado este processo, basta reiniciar o servidor do Apache através da janela Serviços ou do comando Restart, presente no grupo de programas Apache, para que o servidor esteja pronto para a aplicação RIA uniGUI.

Criando a aplicação RIA uniGUI baseada em módulo ISAPI

Para dar início ao desenvolvimento de um simples exemplo baseado em um módulo ISAPI, será criada uma agenda telefônica que possui as funcionalidades de adição e pesquisa de contatos, possibilitando posteriormente a visualização das informações no formato JSON (**Nota do DevMan 1**) , uma vez que não será empregado banco de dados. Com o servidor Apache devidamente instalado e configurado para suportar a aplicação RIA que será utilizada, o próximo passo é iniciar um novo projeto utilizando o assistente do uniGUI através do menu File>New>Other>uniGUI for Delphi>Application Wizard. Na janela do assistente, basta selecionar a opção "ISAPI Module". A **Figura 1** apresenta a janela do assistente do uniGUI para a criação de um módulo ISAPI.

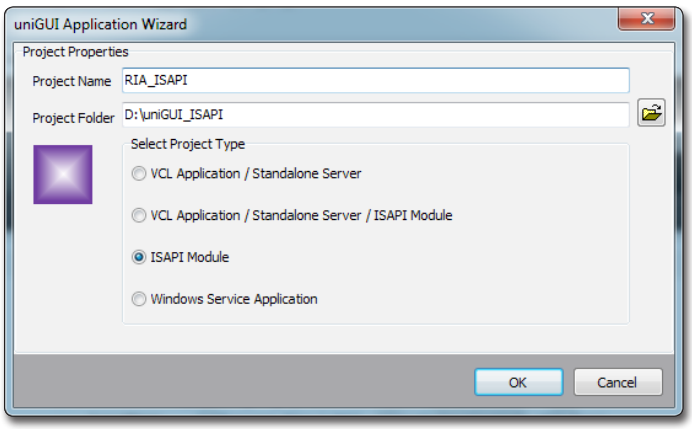

Figura 1. Janela do assistente do uniGUI para a criação de uma aplicação RIA

Nota do DevMan 1

A notação JSON (JavaScript Object Notation) é o padrão adotado para a definição de objetos na linguagem client-side JavaScript, porém não está limitada apenas a esta linguagem, podendo a mesma ser empregada em diversas ferramentas de desenvolvimento, como é o caso do próprio Delphi, através da Unit DBXJSON. Tem sido muito utilizada atualmente para a transferência de informações e armazenamento de dados simples, pois, é mais leve e rápida que o XML, uma vez que é mais enxuta. Sendo assim, o JSON também pode ser empregado em conjunto com AJAX para obter um melhor desempenho nas requisições assíncronas.

Como pode ser observado, além da definição da opção "ISAPI Module", também foram atribuídos o nome para o projeto, RIA_ISAPI, e o diretório "D:\uniGUI_ISAPI", lembrando que o framework VCL também deve ser habilitado para o projeto. Concluída esta etapa, são criados os três objetos básicos para a aplicação uniGUI, o ServerModule, o MainModule e o MainForm, de forma idêntica a um projeto de aplicação autônoma.

O próximo passo é ajustar algumas propriedades no objeto ServerModule, lembrando que ele reúne as principais configurações para a aplicação RIA. Como o uniGUI é baseado na biblioteca

JavaScript ExtJS, para que a aplicação RIA possa funcionar satisfatoriamente, este framework deve sempre acompanhar a mesma. Sendo assim, como o ExtJS foi copiado anteriormente para o mesmo diretório em que o módulo estará presente, a pasta "Sites" configurada no Apache, com o ServerModule selecionado, deve-se alterar a propriedade ExtRoot, substituindo o valor atual "[ext]\", que aponta para o diretório padrão do framework, encontrado na pasta do uniGUI, por "ext-x.y.z\", fazendo com que o mesmo seja carregado no diretório atual. A propriedade LoadingMessage, responsável por exibir a mensagem inicial enquanto a aplicação é carregada, também foi alterada para "Carregando...", assim como a propriedade Title alterada para "RIA", definindo um título para a página principal da aplicação. As subpropriedades presentes em ServerMessages também podem ser alteradas para as outras mensagens da aplicação RIA.

Dando sequência ao projeto, é o momento de adicionar os objetos que serão utilizados ao formulário principal da aplicação (MainForm), lembrando que ao empregar componentes visuais neste tipo de Software, devem-se utilizar os que estão presentes nas paletas do uniGUI. Sendo assim, foram inseridos dois objetos do tipo TUniGroupBox, três TUniLabels, três TUniEdits, três TUniButtons e um TUniDBGrid. Também foram inseridos dois componentes não visuais para gravar temporariamente os dados para o exemplo, sendo um TClientDataSet, para manter os dados na memória, e um TDataSource, para possibilitar a ligação entre os componentes.

No objeto TClientDataSet, que recebeu a denominação de cdsDados, através da propriedade FieldsDef, foram definidos (através de Add New) dois campos do tipo String (DataType), são eles: Nome e Telefone. Para a interligação com os componentes e exibição dos registros, o objeto TDataSource, denominado dsDados, teve a propriedade DataSet definida como cdsDados, bem como o UniDBGrid1 teve a propriedade DataSource definida como dsDados. A **Figura 2** apresenta o visual final do formulário principal da aplicação com os objetos adicionados em tempo de Design.

Terminada a elaboração visual da aplicação, a próxima etapa é iniciar a adição dos códigos. Vale à pena relembrar que a maior parte dos componentes do uniGUI possuem eventos baseados em JavaScript, onde os mesmos rodam diretamente no cliente através do navegador, e a propriedade que permite a definição de tais eventos é a ClientEvents, que por sua vez possui subpropriedades. Sendo assim, será definido um trecho de código para o evento cliente OnClick do botão btSalvar baseado no framework ExtJS. Para tanto, basta selecionar a subpropriedade de ClientEvents, denominada ExtEvents, para abrir a janela de edição de eventos do framework. Em seguida, basta efetuar um clique duplo sob o evento OnClick. A **Listagem 2** apresenta o código que foi definido para este evento.

Neste simples trecho é utilizado o método ajaxRequest para efetuar uma requisição AJAX (Asynchronous JavaScript and XML), que por sua vez invocará o evento interno OnAjaxEvent, que será definido posteriormente, passando como parâmetros o próprio botão (sender), o nome atribuído ao evento e os argumentos para o mesmo. Como pode ser observado, tais argumentos são passados no formato [nome=valor, nome=valor, ...], e o método getValue() é empregado para obter o valor atual do objeto desejado, neste caso os valores digitados nos TUniEdits adicionados anteriormente, acessando os mesmos de acordo com o nome definido. Vale à pena relembrar que o AJAX é uma tecnologia baseada em JavaScript, XML (eXtensible Markup Language) e outros recursos, extremamente importante em aplicações RIA, uma vez que ele permite funcionalidades como requisições assíncronas ao servidor, prévalidações entre outras, o que possibilita um maior desempenho e o aumento da interatividade com o usuário final, dentre outras várias vantagens.

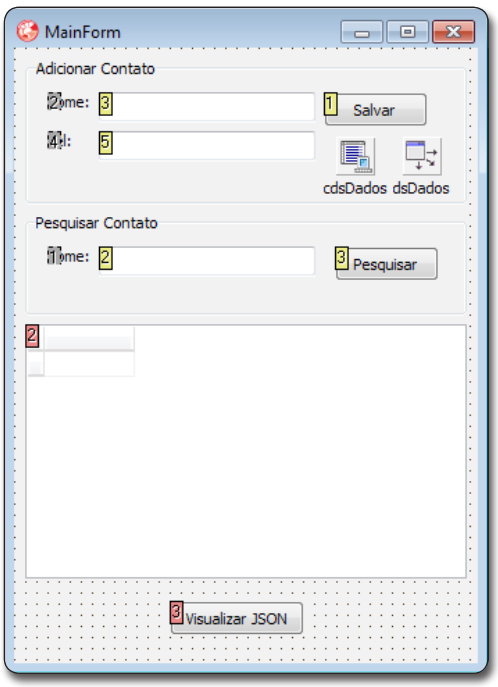

Figura 2. Visual da aplicação em tempo de Design

Listagem 2. Código do evento cliente OnClick do botão btSalvar

```
function OnClick(sender, e)
{
  ajaxRequest(sender,
        'btSalvarClick',
        ['Nome='+MainForm.edNome.getValue(),
         'Tel='+MainForm.edTelefone.getValue()]
  );
}
```

Terminada a definição do evento cliente para o objeto btSalvar, também deve ser adicionado um código para o evento cliente OnClick do outro botão adicionado (btPesquisar) cujo intuito é efetuar uma pesquisa. A **Listagem 3** apresenta justamente o evento OnClick do botão btPesquisar, lembrando que tal código deve ser inserido em ClientEvents.ExtEvents.OnClick.

Como pode ser observado, é semelhante ao código apresentado anteriormente, porém ele invocará o evento OnAjaxEvent do objeto btPesquisar (sender), enviando além do nome, apenas um argumento para o método.

Concluída a definição dos eventos que serão executados no cliente, o próximo passo é implementar os eventos OnAjaxEvent para ambos os botões, que por sua vez efetuará o tratamento das requisições. A **Listagem 4** apresenta o código definido no evento OnAjaxEvent do botão btSalvar.

Listagem 3. Código do evento cliente OnClick do botão btPesquisar

```
function OnClick(sender, e)
{
 ajaxRequest(sender,
       'btPesquisarClick',
       ['Nome='+MainForm.edPesquisaNome.getValue()]
  );
}
```

Listagem 4. Código do evento OnAjaxEvent do botão btSalvar

```
01 procedure TMainForm.btSalvarAjaxEvent(Sender: TComponent; EventName:
    string; Params: TStrings);
02 begin
03   if EventName = 'btSalvarClick' then
04   begin
05     if not cdsDados.Active then
06       cdsDados.CreateDataSet;
07       cdsDados.Append;
08       cdsDados.FieldByName('Nome').AsString  := Params.Values['Nome'];
09       cdsDados.FieldByName('Telefone').AsString := Params.Values['Tel'];
10       cdsDados.Post;
11       edNome.Clear;
12       edTelefone.Clear;
13   end;
14 end;
```

Inicialmente é verificado se o nome do evento cliente que invocou este método é equivalente a "btSalvarClick" (linha 03), conforme definido anteriormente. Este processo é empregado, pois, um mesmo evento OnAjaxEvent pode ser utilizado para manipular outros eventos de cliente, tal como OnMousedown, OnShow e etc, o que torna esta validação um passo importante. Sendo assim, se a condição é satisfeita, então é verificado se o cdsDados, ainda não está ativo para então criar e inicializar o mesmo através do método CreateDataSet, onde evidentemente, o mesmo ficará ativo para as próximas adições (linhas 05 e 06). Em seguida, é iniciada uma nova inserção de registro através do método Append (linha 07). Posteriormente, através de FieldByName, os campos são devidamente preenchidos de acordo com os argumentos que foram enviados pela requisição AJAX, Nome e Tel (linhas 08 e 09), conforme definido anteriormente. Para concluir, o registro é gravado no cdsDados através do método Post (linha 10) e os valores digitados nos objetos TUniEdits são eliminados através do método Clear.

Terminado este processo, o próximo passo é implementar a funcionalidade de pesquisa. A **Listagem 5** demonstra o código atribuído ao evento OnAjaxEvent do botão btPesquisar.

Listagem 5. Código do evento OnAjaxEvent do botão btPesquisar

```
01 procedure TMainForm.btPesquisarAjaxEvent(Sender: TComponent;
    EventName:  string; Params: TStrings);
02 begin
03   if EventName = 'btPesquisarClick' then
04     if cdsDados.Active then
05       cdsDados.Locate('Nome', Params.Values['Nome'],
       [loPartialKey, loCaseInsensitive]);
06 end;
```

Neste trecho de código, também são verificados o nome do evento cliente que invocou o método e o estado do cdsDados (linhas 03 e 04), onde caso sejam satisfeitas as condições, é efetuada uma busca nos registros através do método Locate (linha 05), que recebe como argumentos, o nome do campo para a busca, o valor desejado e dois parâmetros adicionais, loPartialKey, para levar em consideração parte do termo buscado e loCaseInsensitive, para não distinguir entre letras maiúsculas e minúsculas.

Para concluir este exemplo, foi adicionada uma simples funcionalidade para obter o formato em JSON dos contatos gravados pelo usuário. Para empregar os recursos de JSON, a Unit Data.DBXJSON deve ser incluída na cláusula Uses do formulário principal. A **Listagem 6** demonstra o código atribuído ao evento OnClick do botão btVisualizarJson.

Listagem 6. Código do evento OnClick do botão btVisualizarJson

```
01 procedure TMainForm.btVisualizarJsonClick(Sender: TObject);
02 var
03   objeto, contato: TJSONObject;
04   itens: TJSONArray;
05 begin
06   if cdsDados.Active then
07   begin
08     objeto := TJSONObject.Create;
09     cdsDados.First;
10     while not cdsDados.Eof do
11     begin
12       contato := TJSONObject.Create;
13       itens := TJSONArray.Create;
14       contato.AddPair('Nome', cdsDados.FieldByName('Nome').AsString);
15       contato.AddPair('Telefone',
         cdsDados.FieldByName('Telefone').AsString);
16       itens.AddElement(contato);
17       objeto.AddPair('Contato', itens);
18       cdsDados.Next;
19     end;
20     ShowMessage(objeto.ToString);
21   end;
21 end;
```

Inicialmente, são declaradas variáveis do tipo TJSONObject e TJSONArray (linhas 03 e 04). TJSONObject é a classe que permite a implementação de um objeto JSON e a TJSONArray, a que possibilita o emprego de um array. Se o cdsDados está ativo, então o objeto é criado e o cursor é posicionado no primeiro registro do objeto cdsDados através do método First (linhas 08 e 09). Enquanto existirem registros, são criados novos objetos (contato) e um novo

array (itens) (linhas 12 e 13). Em seguida, como a estrutura de um arquivo JSON é constituída basicamente de pares nome:valor, são adicionados pares identificados para o Nome e o Telefone do contato de acordo com o registro atual do cdsDados através do método AddPair (linhas 14 a 17). Posteriormente, o objeto contato é adicionado no array itens (linha 16), que por sua vez é adicionado ao objeto principal (linha 17), indo para o próximo registro através do método Next (linha 18). Para concluir, é apresentado na tela o resultado do objeto JSON, através de sua serialização efetuada pelo método ToString (linha 20).

Neste momento, o desenvolvimento da aplicação RIA baseada em módulo ISAPI está finalizada, restando apenas copiar o arquivo DLL gerado (biblioteca) para a devida pasta configurada no Apache, "Sites". Sendo assim, o Software já pode ser acessado. A **Figura 3** apresenta a aplicação RIA em execução através do Browser.

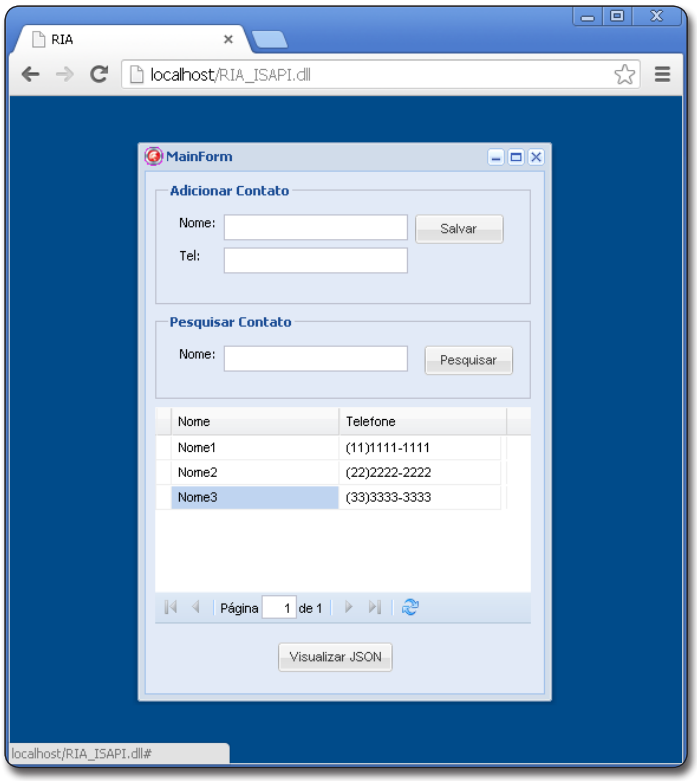

Figura 3. Aplicação RIA baseada em módulo ISAPI em execução

Como foi possível notar, a elaboração de um módulo ISAPI é uma alternativa interessante para ser adotada no desenvolvimento RIA, trazendo diversos benefícios.

Framework Raudus

Notavelmente, apesar de terem a mesma finalidade, o Raudus e o uniGUI apresentam diversas diferenças que vão desde o funcionamento interno e arquitetura até o motor front end, utilizado na elaboração da interface gráfica do usuário (GUI). Como visto, uma aplicação uniGUI é composta basicamente pelos objetos MainForm, MainModule e ServerModule. Já uma aplicação baseada no Raudus, é constituída basicamente por um único formulário. Outra diferença bem perceptível é com relação ao software autônomo, onde no uniGUI, além do mesmo ser executado através de um navegador, ele também pode ser utilizado localmente em um computador por um usuário, como se fosse um sistema Desktop tradicional. Em contrapartida, o Raudus não exibe o formulário principal da aplicação para que o usuário utilize o mesmo como uma aplicação Desktop, e sim uma tela que se assemelha ao Prompt de Comandos do Windows. A **Figura 4** demonstra o executável de uma aplicação Raudus em operação.

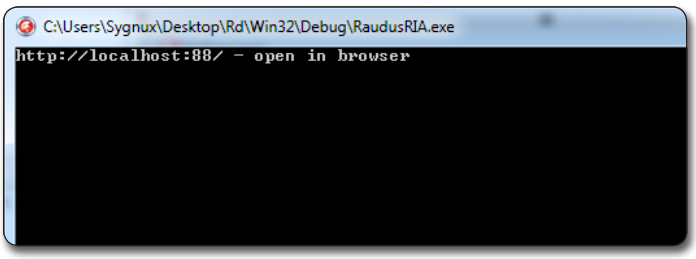

Figura 4. Aplicação Raudus em execução

Como pode ser notado, o software baseado nesse framework deve ser utilizado diretamente pelo browser, através do endereço e do número de porta apresentados (http://localhost:88).

Quanto ao motor front end, o Raudus também diverge do uniGUI, que por sua vez utiliza o framework ExtJS. Em sua versão mais recente, ele possui seu próprio motor Web, uma biblioteca denominada RFE JavaScript, sendo baseado no visual WYSIWYG (What You See Is What You Get – O que você vê é o que você obtém), onde diversos recursos são empregados para que a aparência final da aplicação RIA, a interface gráfica do usuário, seja idêntica ao visual que a mesma possui quando em modo de Design através do Delphi. Indo um pouco mais além, o Raudus também possibilita o emprego de temas visuais para o software. Entretanto, existem outras versões mais antigas que fazem parte desta ferramenta que possuem características diferenciadas, utilizando outros frameworks, como é o caso do Raudus Ext e do Raudus ST. O Raudus Ext também utiliza o ExtJS como motor principal, assim como o uniGUI, sendo mais voltado para as aplicações que serão executadas em computadores. Já o Raudus ST é baseado no Sencha Touch, sendo destinado para a execução em dispositivos móveis.

Todavia, o Raudus também possui semelhanças com o uniGUI. Uma delas é a abstração da camada de codificação HTML e JavaScript, possibilitando que o desenvolvedor utilize apenas a linguagem pascal para elaborar seu projeto. Em contrapartida, ele também permite a adição de codificação desenvolvida manualmente ou de funcionalidades de terceiros para a aplicação RIA, de acordo com a necessidade do projeto, o que também o torna flexível e extensível, não impondo limitações para o software.

Outra característica comum entre ambos é o emprego de AJAX (Asynchronous JavaScript and XML), onde as requisições entre

o cliente e o servidor são realizadas de forma assíncrona, não havendo a recarga completa da página, aumentando assim a performance e a interatividade com o usuário final, condições importantes em uma aplicação RIA.

Contudo, assim como o uniGUI, um dos poucos contratempos que podem ocorrer ao empregar o Raudus é com relação a documentação, que por sua vez é escassa, mas este é um fator que não pesa diante de tantos benefícios que tais frameworks trazem consigo, lembrando ainda que ambos contam com uma diversidade de exemplos que podem ser facilmente estudados e compreendidos.

De fato, escolher qual framework adotar é uma questão da necessidade do projeto, afinal ambas as ferramentas são excelentes alternativas para o desenvolvimento de RIA através do Delphi de uma forma rápida e otimizada.

Instalação do Raudus e os primeiros passos para sua utilização

A instalação do Raudus é extremamente simples. Basta obter e executar o instalador disponibilizado pelo site do projeto (ver seção **Links**) e seguir o tradicional processo Avançar>Avançar>Concluir. Diferente de outros frameworks, não há a necessidade de compilar pacotes ou efetuar configurações adicionais, porém o código fonte do mesmo ainda não está disponível, apesar de ser gratuito.

O Raudus adiciona um novo grupo no Delphi, na janela New Itens. Sendo assim, para iniciar um novo projeto baseado neste framework, basta acessar o menu File>New>Other e selecionar o grupo Raudus. A **Figura 5** apresenta a janela New Itens com este grupo em evidência.

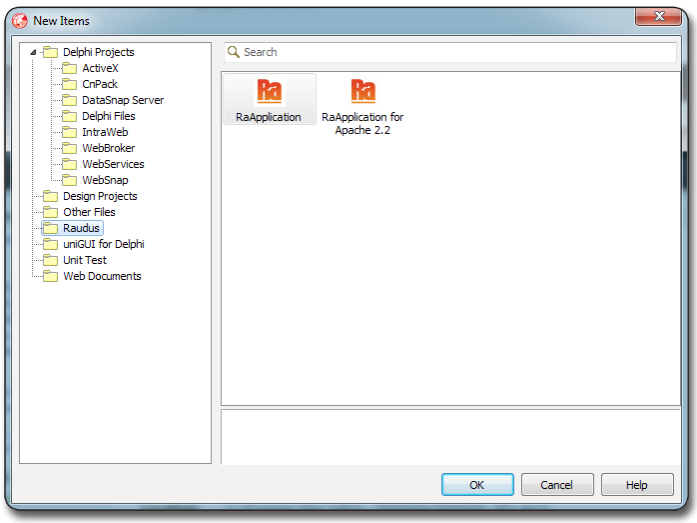

Figura 5. Janela New Itens com o grupo Raudus em evidência

Com a janela New Itens aberta, basta selecionar o item RaApplication, utilizado para a criação de uma aplicação autônoma, confirmar e habilitar o framework VCL para o projeto. Concluída esta etapa, um novo formulário é aberto para que a elaboração da aplicação seja iniciada. Como mencionado anteriormente, neste momento é possível notar que apenas um formulário é criado, diferentemente do uniGUI, que por sua vez adiciona módulos.

Um ponto importante em uma aplicação baseada em Raudus é com relação ao objeto principal da mesma, RaApplication, onde é possível efetuar ajustes gerais para o software, tal como a alteração de propriedades, parâmetros, dentre outros, de forma semelhante em que Application é utilizada para a plataforma Desktop VCL. Outro objeto que pode ser útil neste framework é o RaConfig, que por sua vez permite a definição de diversas configurações para o navegador do usuário.

É importante lembrar que o desenvolvimento de uma aplicação RIA com o Raudus segue o mesmo padrão que a elaboração de um software Desktop, salvo algumas considerações de projeto como o tráfego mínimo de informações para que a aplicação responda rapidamente as solicitações, uma vez que a velocidade da Internet pode comprometer o desempenho.

Criando o banco de dados que será utilizado no exemplo

No exemplo que será apresentado, será elaborada uma aplicação RIA que permite as operações de CRUD (Create, Read, Update and Delete) em uma base de dados. Sendo assim, antes de iniciar o desenvolvimento da mesma, deve ser criado um novo banco de dados utilizando o SGBD Firebird, lembrando que qualquer banco de dados suportado poderia ser empregado, uma vez que os componentes não visuais do Delphi podem ser empregados em conjunto com tais frameworks. Neste banco, será criada uma simples tabela para o cadastro de contatos. A **Listagem 7** apresenta o código SQL empregado na elaboração desta tabela.

Listagem 7. Código SQL para a crição da tabela Contatos

```
CREATE TABLE CONTATOS(
 ID  INTEGER  NOT NULL,
 NOME        VARCHAR(70),
 ENDERECO    VARCHAR(70),
 TELEFONE    VARCHAR(15),
 CONSTRAINT PKY_CONTATOS PRIMARY KEY(ID)
);
```

Neste trecho é possível notar que esta tabela conta com os campos ID, Nome, Endereço e Telefone, tendo como chave-primária o campo ID. Para este exemplo, o banco de dados recebeu a denominação de BASE.FDB.

Criando a aplicação RIA e conectando ao banco de dados

Com um novo projeto Raudus RaApplication criado, no formulário principal da aplicação devem ser inseridos os componentes do framework DBExpress para efetuar a interligação com o banco de dados. Sendo assim, foram adicionados um objeto TSQLConnection, um TSQLDataSet, um TDataSetProvider, um TClientDataSet e um TDataSource, sendo os mesmos renomeados para ConexaoFB, sqlContatos, dspContatos, cdsContatos e dsContatos sucessivamente. O TSQLConnection é o objeto responsável pela conexão com o servidor de banco de dados. O TSQLDataSet é um DataSet unidirecional utilizado neste framework para representar

os registros presentes em uma tabela que foram carregados através de uma instrução SQL do tipo SELECT ou uma procedure. O TDataSetProvider tem por finalidade fornecer os dados presentes em determinado DataSet para um conjunto de dados cliente, sendo também responsável por prover as devidas atualizações para o mesmo. O TClientDataSet possibilita a implementação de um conjunto de dados de forma independente ou integrada a outro DataSet, mantendo os dados na memória. Já o TDataSource é o responsável pela interligação dos DataSets com os objetos utilizados no formulário. Sendo assim, o objeto de conexão, ConexaoFB, deve ser configurado de acordo com o trecho a seguir:

```
ConnectionName = FBConnection
Params.Database = Base\BASE.FDB
LoginPrompt = False
Connected = True
```

Neste trecho, em ConnectionName é definida a conexão com o Firebird (FBConnection), em Params.Database, o endereço do banco de dados, em LoginPrompt com valor False, a tela de Login para o banco de dados não é exibida, e em Connected, a conexão é ativada.

A próxima etapa é configurar o objeto sqlContatos (TSQLDataSet). Sua propriedade SQLConnection deve ser definida com o valor ConexaoFB e sua instrução SQL, atribuída através da propriedade CommandText, deve receber o valor SELECT * FROM CONTATOS. Vale lembrar que convém adicionar os campos deste DataSet através de um clique duplo sob o mesmo e da opção "Add All Fields" presente no menu de contexto. Posteriormente, a propriedade DataSet do objeto dspContatos (TDataSetProvider) deve ser definida com o valor sqlContatos. Em seguida, a propriedade ProviderName do objeto cdsContatos (TClientDataSet) deve receber o valor dspContatos, onde os campos podem ser facilmente carregados efetuando o mesmo processo realizado com o sqlContatos. Para concluir a interligação dos componentes do framework DBExpress, o dsContatos (TDataSource) deve ser ligado ao cdsContatos através de sua propriedade DataSet. Como o cdsContatos é um objeto que mantém as informações na memória, o método ApplyUpdates do mesmo deve ser chamado em dois eventos, AfterPost e AfterDelete, para que as alterações sejam efetivadas no banco de dados. Sendo assim, tais eventos devem receber o trecho de código a seguir:

```
cdsContatos.ApplyUpdates(0);
```

Definido o código para as atualizações, também deve ser atribuído o código para gerar uma ID sequencialmente toda vez que o usuário criar um novo Registro, onde o evento OnNewRecord do cdsContatos pode ser utilizado. A **Listagem 8** apresenta o código responsável por gerar uma ID para o novo registro.

Primeiramente, é declarada uma variável do tipo TSQLQuery, sendo a mesma criada em seguida. Esta classe permite a implementação de uma Query (consulta) utilizando comandos SQL. Também é definida a conexão ativa para a mesma (ConexaoFB). Como pode ser observado, é utilizado um comando contendo a instrução SQL MAX para obter o maior ID e incrementar o mesmo posteriormente para que uma nova ID seja gerada. Para concluir, o objeto Qry é liberado da memória através de FreeAndNil.

Listagem 8. Código para gerar uma nova ID

```
procedure TFoPrincipal.cdsContatosNewRecord(DataSet: TDataSet);
var
 Qry: TSQLQuery;
begin
 Qry := TSQLQuery.Create(nil);
 Qry.SQLConnection := ConexaoFB;
 try
  Qry.SQL.Text := ' SELECT MAX(ID) AS NOVO FROM CONTATOS ';
  Qry.Open;
  cdsContatosID.AsInteger := Qry.FieldByName('NOVO').AsInteger + 1;
  Qry.Close;
 finally
  FreeAndNil(Qry);
 end;
end;
```

Desenvolvendo o visual e os eventos básicos da aplicação RIA

O próximo passo para o projeto é a adição dos componentes visuais que serão utilizados pelo usuário. Como o framework utilizado é o Raudus, os componentes presentes na aba Raudus devem ser empregados para a correta elaboração automática da interface de usuário. Sendo assim, foram adicionados cinco TRaDBEdits, sete TRaButtons, dois TRaPanels, um TRaEdit e um TRaDBGrid. Com os componentes adicionados, a propriedade DataSource do objeto RaDBGrid1 deve ser definida com o valor dsContatos. Para a adição dos campos neste tipo de grid, deve-se utilizar a propriedade Column do mesmo. Com o editor aberto, basta utilizar o botão Add New para inserir todas as colunas desejadas e ajustar a propriedade FieldName de cada coluna adicionada para efetuar o devido vínculo com o banco de dados. Os objetos TRaDBEdits também devem ser vinculados aos devidos campos através do objeto dsContatos. Como os dados serão compartilhados pela Internet ou rede, sempre é importante projetar a aplicação RIA para trafegar o mínimo de informações possíveis para que as consultas e respostas sejam mais rápidas e objetivas. Sendo assim, inicialmente, o DataSet (sqlContatos) ficará ativo, porém, permanecerá vazio até que o usuário informe um determinado termo e clique no botão de pesquisa, trazendo apenas os dados relacionados a busca. Para que o mesmo seja inicializado, foi adicionado no evento OnCreate do formulário principal, um simples trecho de código. Também foram inicializadas algumas configurações para a aplicação RIA. A **Listagem 9** apresenta o código empregado no evento OnCreate.

Neste trecho, inicialmente é definida a mensagem que deve ser exibida enquanto a aplicação Raudus é carregada. Um detalhe interessante é que o desenvolvedor pode empregar as tags em HTML para personalizar a exibição da mesma, tal como no exemplo, onde a tag <font> teve seu atributo color definido

com o valor #9E9E9E, uma cor em hexadecimal, fazendo com a mensagem "Carregando..." tenha sua cor alterada. Também foi utilizado o atributo size, cuja função é definir o tamanho da fonte. Para concluir, o comando SQL do objeto sqlContatos (TSQLDataSet) é alterado para que o mesmo seja aberto logo em seguida.

Listagem 9. Código para inicialização da aplicação RIA

```
procedure TFoPrincipal.FormCreate(Sender: TObject);
begin
 RaApplication.Application.LoadingMessage := '<font color="#9E9E9E"
 size="20">Carregando...</font>';
 sqlContatos.CommandText :=' SELECT * FROM CONTATOS WHERE ID IS NULL ';
 cdsContatos.Open;
end;
```

Neste momento é possível notar que o Raudus permite um desenvolvimento de forma híbrida para a aplicação RIA, onde o desenvolvedor tem a possibilidade de mesclar códigos HTML e JavaScript em conjunto com a linguagem pascal, embora este fator não seja uma obrigatoriedade.

O próximo passo para este exemplo é implementar o código para a pesquisa, utilizando o termo digitado pelo usuário e verificando se existe algum nome cadastrado com o mesmo. Este código foi atribuído ao evento OnClick de um dos botões adicionados anteriormente, btPesquisar. A **Listagem 10** apresenta o código empregado para efetuar a busca.

Listagem 10. Código do evento OnClick do botão btPesquisar

```
procedure TFoPrincipal.btPesquisarClick(Sender: TObject);
begin
 if edPesquisa.Text <> '' then
 begin
  cdsContatos.Close;
  sqlContatos.CommandText :=' SELECT * FROM CONTATOS '+
                  ' WHERE NOME LIKE '+
                  QuotedStr('%'+ edPesquisa.Text +'%');
  cdsContatos.Open;
 end;
end;
```

Como o objetivo da aplicação é trafegar o mínimo de informações, a instrução SQL SELECT utilizando o comando LIKE foi empregada. Também é possível notar que a pesquisa só é realizada caso o usuário tenha digitado alguma informação, lembrando que para a realização desta tarefa, deve-se inicialmente fechar o TClientDataSet através do método Close, definir o novo comando SQL para o objeto TSQLDataSet e em seguida, abrir novamente o TClientDataSet.

Como a aplicação RIA permitirá a adição, edição, gravação e exclusão de registros, tais comandos também devem ser empregados para o objeto cdsContatos, uma vez que até o momento o Raudus não apresenta um componente do tipo TDBNavigator. Sendo assim, o botão btNovo recebeu o comando cdsContatos.Append e o botão btEditar, cdsContatos.Edit. Já os botões btSalvar e btExcluir receberam uma simples validação antes de realizar o comando. O btSalvar deve receber o trecho de código a seguir.

```
if cdsContatos.State in [dsInsert, dsEdit] then
 cdsContatos.Post;
```

Simplesmente, antes de realizar o comando Post, é verificado se o estado do objeto cdsContatos é equivalente a uma inserção ou edição. Já no btExcluir, foi adicionado o trecho seguinte.

```
if cdsContatos.RecordCount > 0 then
 cdsContatos.Delete;
```

Antes de efetuar a exclusão é testado se realmente existem registros carregados para evitar quaisquer falhas.

Criando a funcionalidade de download das informações

Uma funcionalidade extremamente comum e muitas vezes indispensável na Web e também em uma aplicação RIA é disponibilizar para o usuário final opções como download de informações, exportação de dados, etc. O Raudus conta com um componente denominado TRaStreamPublisher. Este componente permite o emprego de um Stream, que pode ser definido como um fluxo de dados com início e fim localizado na memória, como é o caso da classe TStringStream do Delphi, destinada ao emprego de Stream para a manipulação de strings, para exportar a informação desejada em forma de arquivo, dentre outros (**Nota do DevMan 2**). Sendo assim, para deixar a aplicação RIA com maiores funcionalidades, a próxima implementação será disponibilizar para o usuário, a exportação do contato selecionado em forma de XML, adicionando a opção de download. Diante deste fator, foi adicionado ao formulário, um objeto do Tipo TRaStreamPublisher, sendo o mesmo renomeado para RaStreamDownload. Para obter a funcionalidade almejada, a propriedade ContentDisposition, disponibilização do conteúdo, foi definida com o valor attachment, indicando que é um arquivo anexo que pode ser baixado pelo usuário, caso contrário o mesmo seria aberto futuramente em uma nova janela. Também foi definida a propriedade ContentType, tipo do conteúdo, para text/plain (texto). Para possibilitar o download do arquivo XML, o evento OnClick do botão btExportarContato deve receber um trecho de código. A **Listagem 11** demonstra o código deste evento.

Inicialmente, é verificado se existem registros carregados. Através da propriedade DownloadName do objeto RaStreamDownload, é possível definir um nome para o arquivo que será baixado pelo usuário. Na sequência do código, a propriedade Stream deste mesmo objeto recebe uma nova Stream do tipo string (TString-Stream), contendo a estrutura de um arquivo XML com os dados do registro atual obtidos diretamente do objeto cdsContatos. Para finalizar, é adicionado um código HTML para o objeto RaLabel5 através de sua propriedade Caption. Neste código, primeiramente é criada uma divisão, elemento ,

para organizar o conteúdo, alinhando o texto. Posteriormente, é adicionado o elemento <a> e seu atributo href, responsável pela atribuição de um hyperlink, que neste caso serve para informar o endereço de download, onde tal endereço pode ser obtido através da propriedade URL do objeto RaStreamDownload. Também foi adicionado o atributo target com valor _blank, cuja finalidade é abrir uma nova janela não interferindo na atual. Vale à pena ressaltar que as devidas tags empregadas foram devidamente fechadas, . Neste momento, toda vez que o usuário clicar no botão btExportarContato, será criado um novo link no objeto RaLabel5 para que ele possa efetuar o download das informações do contato desejado.

Listagem 11. Código do evento OnClick do botão btExportarContato

```
procedure TFoPrincipal.btExportarContatoClick(Sender: TObject);
begin
 if cdsContatos.RecordCount > 0 then
 begin
  RaStreamDownload.DownloadName := 'Contato.xml';
  RaStreamDownload.Stream :=
  TStringStream.Create(
  '<?xml version="1.0" encoding="utf-8"?>' +
  '<contato>' +
  '<id>' + cdsContatosID.AsString + '</id>' +
  '<nome>' + cdsContatosNOME.AsString + '</nome>' +
  '<endereco>' + cdsContatosENDERECO.AsString + '</endereco>' +
  '<telefone>' + cdsContatosTELEFONE.AsString + '</telefone>' +
  '</contato>'
  );
  RaLabel5.Caption := '<div style="text-align:center;">' +
              '<a href="' + RaStreamDownload.URL +
              '" target="_blank">' +
              'Efetuar Download'+
              '</a>' +
              '</div>';
 end;
end;
```

É importante ter em mente que esta funcionalidade apresentada poderia ser utilizada com os mais diversos fins de acordo com o projeto, já que possibilita a exportação de informações variadas empregando streams, além do que foi demonstrado.

Adicionando a funcionalidade de Impressão

Outra funcionalidade importante que pode ser disponibilizada para o usuário final é a impressão. O mesmo objeto apresentado anteriormente, TRaStreamPublisher, também pode ser utilizado para gerar determinado conteúdo e possibilitar esta tarefa. Como a aplicação cliente está executando em um navegador, para imprimir tal conteúdo, devem ser utilizadas as instruções para impressão neste ambiente, em outras palavras, trechos de código em HTML para obter o visual desejado e comandos/eventos em JavaScript. A **Tabela 1** apresenta uma explicação sobre algumas tags HTML que serão utilizadas na futura elaboração do relatório para este exemplo.

Sendo assim, um novo objeto TRaStreamPublisher foi adicionado ao formulário, onde o mesmo recebeu a denominação de

Nota do DevMan 2

Tipo TStream

A classe base empregada na utilização de Streams é a TStream, permitindo que os dados possam ser lidos, copiados ou gravados para diversos meios, tais como memória dinâmica, memória estática, etc. Também vale a pena citar que esta classe é do tipo abstrata, que por sua vez, é constituída por alguns métodos que não possuem uma definição (virtual) e devem ser elaborados nas classes derivadas, ou em outras palavras, a classe abstrata é a definição de determinado objeto de forma incompleta. Sendo assim, existem outras classes que descendem de TStream, onde cada uma é direcionada a manipulação de determinado tipo de dado e implementam funções mais especificas para seu objetivo:

TFileStream: Destinada à manipulação de arquivos, permitindo que as aplicações possam ler e gravar os mesmos em um meio físico;

TMemoryStream: Empregada para a manipulação de dados na memória dinâmica, capacitada também a oferecer recursos de I/O (Entrada e Saída);

TStringStream: Destinada à manipulação de Strings, oferecendo recursos de I/O e permitindo a gravação das informações diretamente para um meio físico;

TOleStream: Utilizada para interagir com uma interface COM para escrita e leitura de um objeto OLE;

TBlobStream: Empregada em manipulações com campos do tipo BLOB, ou em outras palavras, grandes objetos do tipo binário;

TWinSocketStream: Utilizada para escrever e ler informações através de uma conexão de soquete (Socket);

THandleStream: Destina-se à leitura e gravação de soquetes, arquivos, entre outros, empregando um identificador (Handle).

As classes derivadas de TStream possuem diversos métodos para que o desenvolvedor possa empregá-los em diferentes momentos, podendo manipular os dados, alguns comuns entre todas as subclasses e outros mais específicos. Outro ponto importante relacionado à manipulação de Streams é que ao adotar este recurso, o desenvolvedor pode através de sua aplicação, dividir a informação em partes, gravando-as de forma separada, ou ainda unir uma informação a outra. A seguir temos alguns dos principais métodos empregados na criação e manipulação de Streams.

Create: O método construtor deve ser utilizado para a criação de uma instância da classe em questão. Na classe TFileStream, alguns parâmetros podem ser especificados, tais como o fmCreate, indicando que o arquivo deve ser criado ou substituído;

SetSize: Permite a definição do tamanho (Bytes) de um Stream;

Read: Utilizado para a leitura dos dados de um Stream para um Buffer, retornando o número de Bytes lidos;

Write: Utilizado para a escrita de dados presentes em um buffer para um Stream, retornando o número de Bytes gravados;

Seek: Define a posição atual para um Stream, permitindo assim se mover pelo mesmo;

CopyFrom: Empregado para copiar os dados de um Stream para outro de uma forma simples;

ReadBuffer: Efetua a leitura dos dados de Stream para um Buffer;

WriteBuffer: Efetua a gravação dos dados de Stream para um Buffer.

Além dos métodos, as classes descendentes de TStream possuem algumas propriedades que são importantes, tais como Size, que armazena o tamanho atual de um Stream, e Position, que indica a posição atual.

Contudo, em um bloco de código podem ser empregadas instâncias de classes diferentes derivadas de TStream que podem compartilhar informações entre si, em outras palavras, um objeto derivado de TFileStream por exemplo, pode ler os dados presentes em um outro objeto que foi derivado da classe TStringStream, e assim gravá-los efetivamente em um meio físico.

RaStreamImpressao. Para implementar a funcionalidade de impressão, este objeto teve apenas a propriedade ContentType definida com o valor text/html, informando que o conteúdo está em HTML. Para criar um simples relatório dos contatos atuais carregados e possibilitar a impressão do mesmo, no evento OnClick do botão btImprimirContato, deve ser adicionado um trecho de código. A **Listagem 12** demonstra o código atribuído a tal evento.

De forma semelhante ao processo de download, primeiramente é verificado se existem registros carregados no cdsContatos. Em seguida, foi criada uma variável do tipo string com o intuito de auxiliar no processo de elaboração do relatório, denominada html. O cursor do cdsContatos também foi posicionado para o primeiro registro (First). Da linha 09 até a linha 11, são definidas as tags iniciais para o documento HMTL. Da linha 12 até a linha 19, tag <style>, são criadas classes (.principal, .tabela, etc) contendo informações de formatação para os elementos que pertençam as mesmas, possibilitando a definição de margens, bordas, cores, fontes, etc. Da linha 20 até a linha 25 é criada uma simples função JavaScript, Imprimir, onde primeiramente é definido um tempo de um segundo (linha 22) para que o comando responsável pela

Listagem 12. Código do evento OnClick do botão btImprimirContato

```
01 procedure TFoPrincipal.RaButton1Click(Sender: TObject);
02 var
03  html: String;
04 begin
05  if cdsContatos.RecordCount > 0 then
06 begin
07  cdsContatos.First;
08  html :=
09  '<!doctype html><html>' +
10  '<head>' +
11  '<title>Impressão de Contatos</title>' +
12  '<style>' +
13  '.principal {margin: 10pt; border: solid black 1px;}' +
14  '.principal td {padding: 2pt; font: normal 12px/20px arial; text-align: center;
      vertical-align: top;}' +
15  '.principal td b {font: bold 15px/30px arial; color: black;}' +
16  '.tabela {margin: 0; border-spacing: 0; border-collapse: collapse;}' +
17  '.tabela td {border: solid black 1px; text-align: center; font: normal 12px/20px arial;
      color: black;}' +
18  '.tabela td b {font: bold 15px/30px arial; color: black;}' +
19  '</style>' +
20  '<script type="text/javascript">' +
21  'function Imprimir() {' +
22  '  setTimeout(function(){window.print();},1000);' +
23  '  setTimeout(function(){this.onmousemove=function(){window.close();};},1000);' +
24  '}' +
25  '</script>' +
26  '</head>' +
27  '<body onload="Imprimir()">' +
28  '<center><div style="page-break-inside: avoid;">' +
29  '<table class="principal" width="90%"><tbody>' +
30  '  <tr>' +
31  '    <td width="80%"><b>Impressão de Contatos</b></td>' +
32  '    <td width="20%" style="border-left: solid black 1px;">Página 1 de 1</td>' +
33  '  </tr>' +
34  '</tbody></table>' +
35  '<table class="tabela" width="90%"><tbody>' +
36  '  <tr>' +
37  '    <td><b>ID</b></td>' +
38  '    <td><b>Nome</b></td>' +
39  '    <td><b>Endere&ccedil;o</b></td>' +
40  '    <td><b>Telefone</b></td>' +
41  '  </tr>';
42  while not cdsContatos.Eof do
43  begin
44  html := html +
45  '  <tr>' +
46  '    <td>' + cdsContatosID.AsString + '</td>' +
47  '    <td>' + cdsContatosNOME.AsString + '</td>' +
48  '    <td>' + cdsContatosENDERECO.AsString + '</td>' +
49  '    <td>' + cdsContatosTELEFONE.AsString + '</td>' +
50  '  </tr>';
51  cdsContatos.Next;
52  end;
53  html := html +
54  '  <tr>' +
55  '    <td colspan="3" style="padding: 0 20px 0 0; text-align: right;">
       <b>Total de Registros: </b></td>' +
56  '    <td><b>' + IntToStr(cdsContatos.RecordCount) + '</b></td>' +
57  '  </tr>' +
58  '</tbody></table>' +
59  '</div></center>' +
60  '</body></html>';
61  RaStreamImpressao.Stream := TStringStream.Create(html);
62  RaLabel6.Caption :=
63  '<div style="text-align:center;line-height:25px;">' +
64  '<a href="' + RaStreamImpressao.URL + '" target="_blank">' +
65  'Imprimir Contato' +
66  '</a>' +
67  '</div>';
68  end;
69 end;
```

Tag HTML	Descrição
<html></html>	Denota o início e o fim de um documento baseado em HTML.
<head></head>	Uma das tags básicas para o documento, empregada para a definição de propriedades, estilos, scripts, relação com arquivos externos, dentre outros.
<body></body>	Tag base utilizada para a definição do corpo do documento HTML, devendo conter os elementos visuais para o mesmo.
<style></style>	Empregado para a definição de estilos permitindo uma melhor formatação e organização dos elementos do documento, em outras palavras, folhas de estilo em cascata (CSS).
<script></script>	Utilizado para a adição de scripts para o documento, tal como códigos na linguagem JavaScript.
	Permite a criação de um container/área para organizar o documento.
<table></table>	Define uma tabela.
<tr></tr>	Define uma linha de uma tabela.
<td></td>	Define uma coluna de uma tabela.

Tabela 1. Algumas tags HTML que serão empregadas na elaboração do relatório

impressão, window.print, seja invocado. Vale à pena evidenciar que este comando não iniciará a impressão do documento imediatamente, mas exibirá a janela de pré-visualização do navegador em questão para esta finalidade. Na linha 23 também é definido um tempo de um segundo para disparar o evento onmousemove (ao movimentar o mouse sobre o documento) empregando o comando window.close, cuja finalidade é fechar a janela contendo o documento de impressão automaticamente. A partir da linha 27 começa a ser definido o visual do documento. Como pode ser notado na linha 29, é criada uma tabela com 90% da largura disponível para apresentar o título do relatório. Em seguida, é criada uma nova linha para a tabela (linha 30) e duas colunas, uma contendo o título e outra contendo um número de páginas (linhas 31 e 32). Na linha 35, a tabela é devidamente fechada e logo em seguida, é criada uma nova tabela para o conteúdo. Da linha 36 até a linha 41, é criada uma nova linha para a tabela contendo quatro colunas, utilizadas para apresentar os títulos, ID, Nome, Endereço e Telefone. Em seguida, na linha 42, é efetuado um laço do tipo while até o final dos registros do cdsContatos. O objetivo deste laço é que, para cada registro encontrado, seja criada uma nova linha e quatro novas colunas para a tabela do documento HTML contendo as informações do registro atual. Posteriormente, da linha 53 até a linha 60, é efetuado o fechamento da tabela possuindo duas colunas e apresentando o número total de registros encontrados. Na linha 61, a propriedade Stream do objeto RaStreamImpressao recebe um novo TStringStream criado com base no valor da variável html. Para concluir, na linha 62, é adicionado o hyperlink contendo o endereço para que a impressão possa ser realizada.

Alterando o visual dos componentes com a utilização de temas

O último passo para o desenvolvimento deste exemplo é alterar o tema dos componentes visuais que fazem parte da aplicação, para que a mesma não permaneça com o visual clássico. Grande parte destes componentes do framework Raudus possuem uma propriedade denominada UI, que por sua vez contém alguns temas que podem ser aplicados exclusivamente para aquele objeto. Neste projeto, ambos os painéis adicionados tiveram a propriedade UI definida com o valor cupertino-grad-v, aplicando um visual com efeito gradiente. Já para os botões e os Edits, o tema selecionado foi o cupertino, onde as bordas se tornam arredondadas. Neste momento, a aplicação RIA deste exemplo está concluída.

Executando a aplicação Raudus

Após executar a aplicação RIA, basta abrir um navegador e acessar a URL http://localhost:88. É importante lembrar que localhost aponta para a interface local de rede, 127.0.0.1, portanto, para que o software possa ser acessado pelas demais máquinas da rede, o endereço IP real da mesma deve ser utilizado, tal como o exemplo http://192.168.1.1:88. Outro fator relevante é com relação ao ambiente da Internet, onde toda a estrutura da rede interna deve ser levada em consideração para que a aplicação RIA possa ser acessada remotamente, uma vez que podem existir Firewalls, Gateways, Roteadores, etc, que por sua vez podem barrar o acesso direto à mesma. Diante deste fato, deve-se liberar/redirecionar a porta utilizada pela aplicação RIA baseada em Raudus (88 ou outra definida) nos devidos dispositivos da rede para que a mesma possa ser utilizada. A **Figura 6** apresenta o visual final da aplicação.

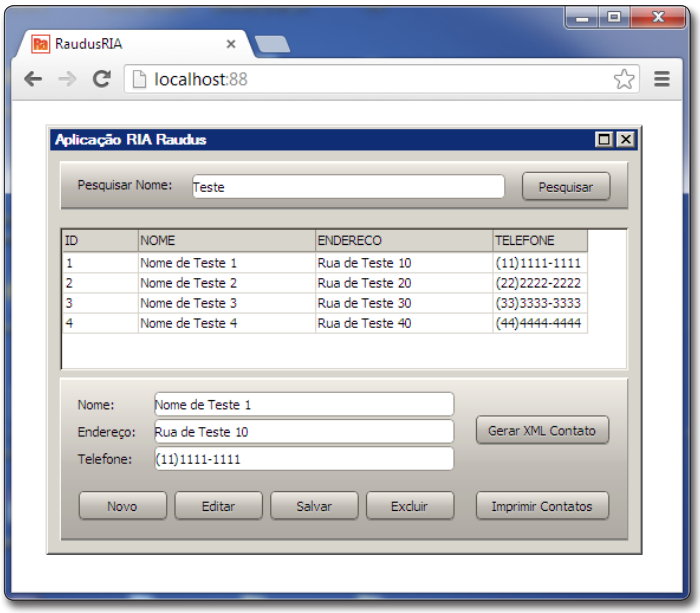

Figura 6. Aplicação RIA baseada em Raudus personalizada em execução

Como pode ser observado, após a definição dos temas, a aplicação ficou com um visual diferenciado, com exceção do formulário que foi mantido o tema clássico. A **Figura 7** demonstra o visual final do relatório para impressão.

De fato, o Raudus é um excelente framework muito útil para elaboração de aplicações RIA que traz consigo diversos componentes e recursos que possibilitam atender as necessidades de um software comercial.

Raudus para clientes na plataforma móvel

Como mencionado anteriormente, o Raudus pode ser facilmente empregado para o desenvolvimento de uma aplicação RIA cujo foco é a plataforma móvel, uma vez que conta com temas visuais que podem ser utilizados na mesma. Para elaborar uma aplicação com esta finalidade, basta iniciar um novo projeto Raudus autônomo (RaApplication). Este framework traz consigo uma Unit denominada RxTheme.pas que possibilita a alteração do tema padrão dos formulários. Sendo assim, basta copiá-la e adicioná-la ao projeto, lembrando que a mesma pode ser acessada através do diretório C:\Users\<Nome do Usuário>\Documents\Raudus\examples\20-RxTheme. Em seguida, basta incluir a Unit RxTheme no formulário atual e declarar uma variável do tipo TRxTheme, tal como RxTheme1. Para concluir a aplicação do tema para o formulário, basta inserir no evento OnCreate do mesmo, o trecho de código a seguir:

```
RxTheme1 := TRxTheme.Create(Self);
RxTheme1.ForceMobile := True;
```

Sendo assim, basta inserir os outros objetos que farão parte da aplicação, tal como TRaPanels, TRaButtons, TaEdits, e definir um tema para os mesmos através da propriedade UI, tal como cupertino. A **Figura 8** apresenta um visual que pode ser obtido com o auxílio da Unit RxTheme.

Como pode ser observado, o Raudus possibilita a personalização da aplicação de uma forma bem simples, empregando os recursos que por padrão, já fazem parte do framework.

Vale a pena recordar também que o Raudus possibilita a criação de um módulo que pode ser carregado pelo servidor Apache (site.so), simplesmente utilizando a opção RaApplication for Apache 2.2 da Janela New Itens.

Impressão de Contatos			Página 1 de 1
ID	**Nome**	**Endereço**	**Telefone**
1	Nome de Teste 1	Rua de Teste 10	(11)1111-1111
2	Nome de Teste 2	Rua de Teste 20	(22)2222-2222
3	Nome de Teste 3	Rua de Teste 30	(33)3333-3333
4	Nome de Teste 4	Rua de Teste 40	(44)4444-4444
		Total de Registros:	**4**

Figura 7. Visual do relatório de contatos que pode ser impresso

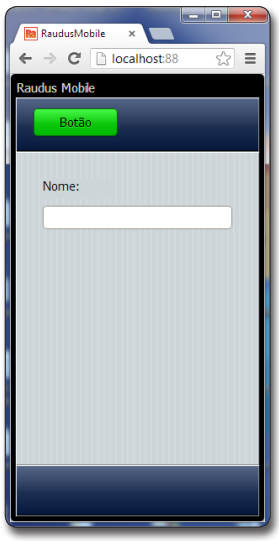

Figura 8. Aplicação RIA com tema para a plataforma móvel

Conclusão

Como foi demonstrado neste artigo, os servidores HTTP como Apache e IIS são muito populares no ambiente da Internet, logo se tornam importantes bases para a disponibilização da aplicação RIA de uma forma diferente, mais compatível e abrangente, que como visto, é possível através da criação de módulos ISAPI que podem ser facilmente carregados, atualizados e migrados de servidores sem problemas.

A Internet realmente é uma plataforma com um grande potencial. Neste ambiente, graças as suas padronizações, podem ser desenvolvidas aplicações multicamadas capacitadas a atender os mais variados ambientes operacionais, a qualquer momento e em qualquer lugar que exista disponibilidade de acesso. Com frameworks como o uniGUI e o Raudus, o desenvolvimento de uma aplicação RIA se tornou uma tarefa extremamente rápida e simples, utilizando a própria linguagem pascal, componentes do próprio Delphi e todas as vantagens do ambiente RAD, de fato, um grande passo na elaboração deste tipo de aplicações, afinal, estes frameworks empregam suas ferramentas para gerar toda a interface de usuário de forma automática.

Como demonstrado, além de todas as vantagens que estes frameworks proporcionam, eles ainda podem ser estendidos, onde o desenvolvedor pode empregar outras bibliotecas e APIs de terceiros que necessite, ou ainda criar novas funcionalidades através de códigos em HTML e JavaScript. Qual deles escolher é algo que vai depender do projeto ou do desejo do desenvolvedor, pois ambos são excelentes ferramentas de trabalho.

Os exemplos apresentados exploram algumas das funcionalidades mais comuns e necessárias destes frameworks no dia-a-dia, em aplicações comerciais, por exemplo, e de forma bem simples, onde é importante ressaltar que ambos possuem ainda mais funcionalidades e recursos do que foi demonstrando.

Giuliano Scombatti Pinto
giuliano@sygnux.com.br
É Analista Desenvolvedor da Sygnux Software (www.sygnux.com.br) localizada em Monte Alto/SP, atuando também com desenvolvimento Web e Android. Formado pela Fatec/TQ, começou a desenvolver com Delphi em 2003. Atualmente trabalha com Delphi, Java, PHP e Flash/Flex. Professor de Informática de Projetos Sociais da Prefeitura Municipal de Monte Alto/SP.

Maiores informações e download do Apache
http://www.apache.org

Maiores informações e download do uniGUI
http://www.unigui.com

Maiores informações e download do Raudus
http://www.raudus.com

Maiores informações e download do ExtJS
http://www.sencha.com/products/extjs/

Maiores informações sobre a API JavaScript do Google Maps
https://developers.google.com/maps/documentation/javascript/reference

Dê seu feedback sobre esta edição!

A ClubeDelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/clubedelphi/feedback

Modelagem UML para Projeto não Orientado a Objetos

Principais diagramas a utilizar

Modelar um software tem o objetivo de tornar mais clara a visualização de suas funcionalidades, componentes, interações, seja para um desenvolvedor ou para um cliente. Ao invés de tentar criar seus próprios símbolos ou identificadores, porque não utilizar algo já consistente e que é um padrão internacional? Esse padrão é a UML.

Temos no Delphi XE3 recursos disponíveis para realizar essa modelagem, não é necessário correr atrás de outro software, tudo o que é necessário encontra-se no IDE. Mas antes de modelar algo, temos um histórico resumido sobre a UML.

Um breve histórico da UML

Sua sigla significa Unified Modeling Language ou Linguagem de Modelagem Unificada. É preciso deixar bem claro que ela não é uma metodologia, processo ou mesmo linguagem de programação. É uma forma padronizada de notação.

Durante os anos 90 existiam vários métodos coexistentes que conflitavam entre si a cerca das notações que utilizavam. Os mais conhecidos eram:

- OMT – Object Modeling Techinique, de Rumbaugh;
- Método de Booch;
- OOSE – Object Oriented Software Engineering, de Jacobson.

Em 1995 Booch e Rumbaugh uniram seus processos e notações, eles trabalhavam juntos na Rational Software, que hoje é uma divisão da IBM, e dessa fusão criou-se o Método Unificado. Posteriormente, Jacobson integrou-se a eles, o que resultou em um novo método chamado RUP – Rational Unified Process. Suas notações também foram unidas, gerando a UML.

Resumo DevMan

Modelagem UML para Projeto não Orientado a Objetos:

Fazer a representação visual de um sistema, ou de uma ideia nem sempre é fácil. Existe uma linguagem apropriada para expressar comportamentos, relações, funcionalidades em um sistema orientado a objetos. A UML. Mas, esse é o ponto, ela possui características específicas para a orientação a objetos. Muito desenvolvedor Delphi foge da UML por achar que, por não utilizar a orientação a objetos em sua totalidade, não precisa modelar ou até mesmo que ela não lhe será útil. Muitos preferem criar um "rabisco" aqui e outro ali e quando não, criar longos manuais descritivos e até mesmo com figuras. Toda essa documentação é criada para facilitar a integração de novos desenvolvedores, para tornar clara a visão de um sistema. O uso de uma linguagem padrão (UML) tem suas vantagens. Por ser um padrão, sua representação não irá variar de uma empresa para outra, assim, um desenvolvedor que já a conheça irá facilmente compreender os diagramas criados e se inteirar da lógica envolvida. Mesmo com aspectos ligados ao paradigma da orientação a objetos, existem alguns diagramas que podem ser utilizados mesmo em softwares procedurais. Neste artigo vamos abordar alguns diagramas que se adequam perfeitamente a essa situação e através de um projeto bem simples, mas bem tradicional, vamos modelar sua funcionalidade de tal forma que as vantagens da UML ficarão claras. São diagramas simples que irão melhorar a comunicação entre sua equipe, irão facilitar o entendimento dos requisitos envolvidos e que irão agregar valor à análise do negócio.

Em que situação o tema é útil:

Quando se deseja documentar, coletar requisitos para desenvolvimento de funcionalidades de um software ou quando se deseja compartilhar as regras de negócio e implementações envolvidas em um projeto.

Pode-se assim dizer que a UML é o resultado da busca pela padronização dos artefatos de análise e projeto, quais sejam: modelos semânticos, sintaxe de notações e diagramas. Sua importância está nos seguintes fatos:

• Serve como linguagem para expressar decisões que nem sempre são fáceis de ver apenas olhando para o código;

• Fornece semântica para capturar decisões estratégicas e táticas do projeto;

• É uma forma concreta para o entendimento das pessoas e para ser manipulada por máquinas;

• Não possui dependência com linguagens de programação e métodos de desenvolvimento.

Sua composição

A UML possui um conjunto de notações para criar modelos visuais de um determinado software, combinando para isso modelagem de dados, negócios, objetos e componentes. Seu objetivo é descrever o que fazer, como fazer, quando fazer e porque deve ser feito, contudo, sem estar arramado a tecnologia alguma. Por isso é dito que a UML pode ser utilizada qualquer tipo de projeto.

Seus diagramas são usados de forma combinada para que seja possível visualizar os mais variados aspectos de um sistema, assim, eles são organizados em Estruturais e Comportamentais.

- Diagramas Estruturais
  - Diagrama de classe
  - Diagrama de objeto
  - Diagrama de componentes
  - Diagrama de implantação
  - Diagrama de pacotes
  - Diagrama de estrutura composta

- Diagramas Comportamentais
  - Caso de uso
  - Máquina de Estados
  - Diagrama de Atividades
  - Diagramas de interação
    - Sequência
    - Interação geral
    - Comunicação
    - Tempo

Os diagramas

É definido um grande número de diagramas que se dirigem aos mais variados focos de um sistema, onde, se bem usados, facilitam a compreensão do software que está em desenvolvimento. A seguir temos uma descrição sucinta deles e mais adiante, temos detalhes dos principais diagramas.

• **Diagrama de casos de uso** - É utilizado para especificar um conjunto de funcionalidades e os agentes externos que interagem com o sistema. Exibe uma visão estática dos requisitos.

• **Diagrama de classes** – Seu resultado é o mais próximo da estrutura de código de um programa, mostrando o conjunto de classes, métodos, atributos e relacionamentos entre classes.

• **Diagrama de Objetos** – É uma variação do diagrama de classes, que mostra as instâncias (objetos) em um determinado ponto no tempo. São exibidos esses objetos e seus relacionamentos.

• **Diagrama de Pacotes** – Funciona como um agrupador que une e organiza conjuntos de classes, diagramas, arquivos, ou outros pacotes. Sua finalidade é particionar a estrutura de forma lógica, descrevendo as dependências dentre esses pacotes. Pode ser utilizado para representar a arquitetura de um sistema.

• **Diagrama de Sequência** – Enfatiza a ordem temporal de troca de mensagens entre objetos de um sistema. Por mensagens entendam-se chamadas de funções, métodos, procedimentos. Esse diagrama mostra a colaboração entre objetos em uma determinada funcionalidade do sistema, isso dá ao desenvolvedor um entendimento sobre o que acontece em determinado ponto do tempo.

• **Diagrama de Comunicação** – É um complemento do diagrama de sequência, mostrando como os objetos estão vinculados através das mensagens. Aqui, o tempo não é modelado, mas a ordem da troca de mensagens é definida.

• **Diagrama de Atividades** – É muito parecido com um DFD (diagrama de fluxo de dados), mostrando a execução das ações e a movimentação realizada pela conclusão dessas. Esse diagrama está sempre relacionado a um caso de uso, permitindo descrever ações sequenciais ou paralelas.

• **Diagrama de Estrutura Composta** – Detalha a composição de uma funcionalidade em termos que mostram como os objetos, classes envolvidas, colaboram entre si para concluir a funcionalidade.

• **Diagrama de Componentes** – Mostra como as classes se encontram organizadas em componentes de trabalho.

• **Diagrama de Implantação** – Também é conhecido como diagrama de instalação. Descreve e organiza tudo o que é necessário para um software funcionar, como componentes de hardware, software e outros objetos necessários.

• **Diagrama de Tempo** – Modela o comportamento de objetos e sua interação ao longo de um tempo, mostrando o que é mudado nesse período. Em resumo, representa os possíveis estados de um objeto frente a eventos diversos.

•**Diagrama de Máquina de Estados** – Pode ser utilizado para representar os estados de um caso de uso, subsistema ou um sistema completo. Ele mostra a situação (estado) de um elemento.

•**Diagrama de Interação Geral** – Apresenta uma visão geral de um sistema, subsistema ou processo de negócio. É uma variação do diagrama de atividades.

Delphi XE3 e a UML

Como foi dito, a UML é utilizada para modelagem, em especial, de software orientado a objeto. Contudo, essa não é a realidade da grande maioria dos desenvolvedores Delphi, que acabam optando pelo conceito RAD, criando aplicações centradas a dados. Fica a pergunta, é possível utilizar UML para representar um sistema "padrão"?

O Delphi apresenta suporte à modelagem UML, de forma nativa na versão Architect. Para demonstrar como essa funciona-

lidade pode ser aplicada em sistema que não sejam totalmente orientados a objetos e criar alguns dos principais diagramas, foi criada uma nova aplicação do tipo VCL (File>New>VCL Forms Application). Em seu formulário principal foram dispostos controles de modo a simular um cadastro real de pessoas, como mostra a **Figura 1**.

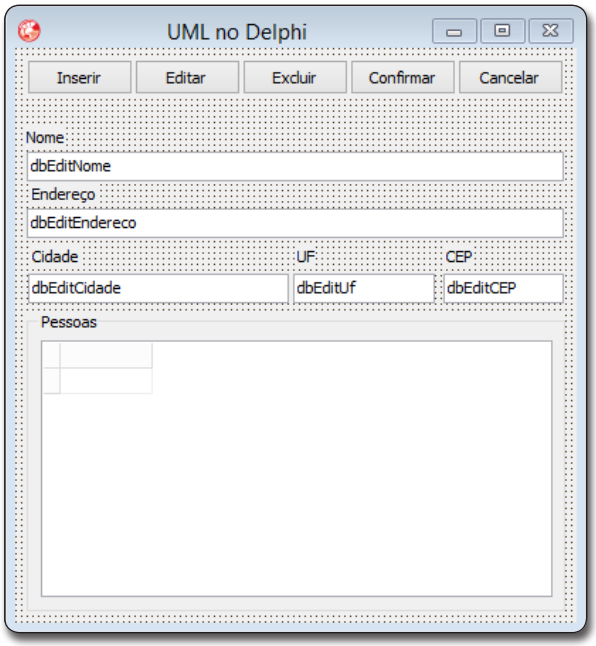

Figura 1. Tela de cadastro

Os botões de operação foram renomeados para representar sua funcionalidade, por exemplo, o botão Inserir possui o nome btnInserir e assim por diante. Isso é uma forma de documentação, pois auxilia na identificação das responsabilidades dos controles visuais. Na sequência, foi adicionado ao projeto um Data Module de nome dtm. Nele foi inserido um componente TClientDataset, que será nosso repositório temporário de informações. Nessa aplicação vamos manter os dados em memória apenas, visto que nosso foco está na modelagem em si. Nesse TClientDataSet, chamado de cdsClientes, foram adicionados campos que pudessem conter os dados exibidos pela **Figura 1**. Posteriormente uma referência desse Data Module foi incluída no formulário principal e os controles Dataware foram devidamente conectados.

Mas como sabíamos quais eram os campos necessários? Como saberemos quais são as restrições de negócio desse cadastro? Isso faz parte da análise de requisitos.

Modelando requisitos

Para coletar informações de como um determinado recurso deve funcionar, no ponto de vista do usuário, utilizamos um diagrama de caso de uso. Assim, para adicionar esse ao projeto é preciso acessar a guia Model View do Project Manager (**Figura 2**). Ao clicar sobre ela, o Delphi nos pergunta se queremos ativar o suporte de modelagem ao projeto, respondemos então que sim.

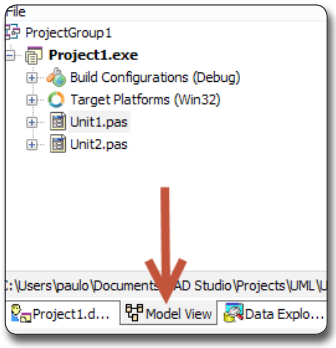

Figura 2. Visualização de modelagem no Project Manager

A partir da visualização de modelagem podemos clicar com o botão direito sobre o item que representa o projeto (project1) e em seu sub menu escolher Add>Other Diagram... Essa opção faz com que seja aberta uma janela para selecionarmos qual tipo de diagrama vamos incluir no projeto. É possível notar (**Figura 3**) que temos alguns dos diagramas citados. Dessa forma então escolhemos Use Case Diagram, dando-lhe o nome de "Caso de Uso – Cadastro".

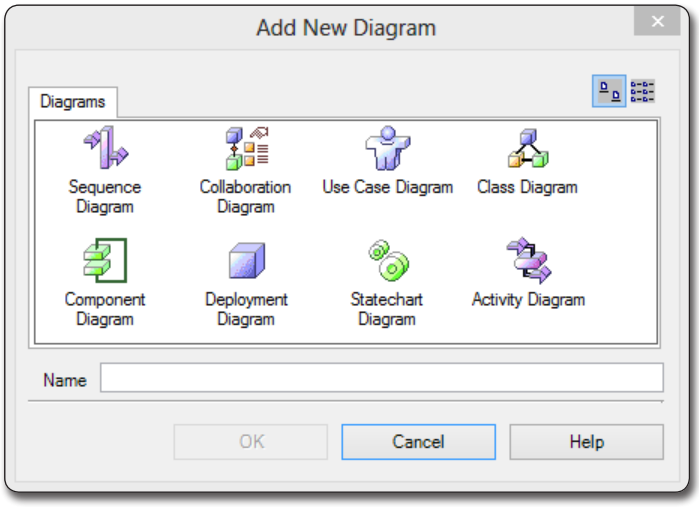

Figura 3. Opções disponíveis de diagramas

É importante ressaltar que a tarefa de extração de requisitos para um caso de uso é utilizada para determinar os serviços que o usuário espera do sistema e suas condições. Assim, o analista deve ser capaz de reconhecer dois tipos de requisitos: os funcionais e não funcionais.

Em suma, um requisito funcional representa um comportamento do sistema. Nele constam informações de entrada, o processamento e a saída esperada. O não funcional indica aspectos como performance (a consulta não pode demorar mais que 5 segundos), segurança (é preciso ter um controle de usuários), identificação visual (cores, temas, posicionamento dos controles na tela) e outros mais. Em tese, um caso de uso contém diretamente os requisitos funcionais de um software.

Como um Caso de Uso é um artefato de comunicação, entre clientes, usuários e desenvolvedores, é fácil de ser compreendido e simples de ser representado. Assim, ele é composto por alguns elementos básicos: atores, casos de uso e relacionamentos.

Um ator é representado por um boneco com um nome, representando algo que interage com a rotina do sistema. Para o caso de uso temos uma elipse que contém o nome do caso de uso em si. Esse apresenta uma função no sistema. Já os relacionamentos indicam como esses dois primeiros elementos interagem. Entre ator e caso de uso temos um relacionamento de associação, indicando que o Ator executa a ação definida pelo caso de uso. Entre atores também é possível haver generalização, indicando que um ator herda características de outro. Entre casos de uso temos o tipo Include, quando um caso de uso A indica para um caso de uso B que B é essencial para o comportamento de A. Também é possível termos o tipo de relacionamento Extend, que é um pouco complexo, mas ele diz que um caso de uso B pode ser acrescentado para descrever um caso de uso A, mas não é necessário. É como se B fosse uma variação de A que é executada em situações específicas e em pontos específicos. Também é possível haver generalização entre casos de usos.

Dessa forma, a **Figura 4** mostra os elementos que o Delphi nos disponibiliza na Tool Palette para a criação do caso de uso.

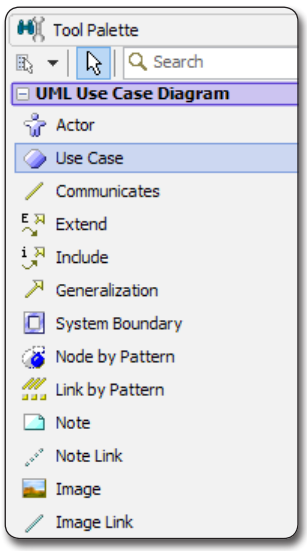

Figura 4. Elementos de Caso de Uso

Vemos que há outros elementos além do que foi comentado. O trabalho de modelagem agora consiste em arrastar esses elementos de forma que representem o que deve ser feito. A **Figura 5** mostra o resultado final.

Foi utilizado um elemento de nota explicativa para dar um detalhe a mais sobre o caso de uso. Mas, ainda faltam mais informações, por exemplo, que informações compõem um cliente? Como deve ser a reação do sistema ao encontrar um nome vazio?

Tudo isso fica relacionado da documentação de Caso de Uso. Isso é necessário porque o diagrama é apenas uma visualização, um panorama de funcionalidades. A documentação deve constar informações que identifiquem qual caso de uso ela se refere, uma descrição geral do caso de uso, o ator principal, pré-condições para a execução da funcionalidade, o fluxo principal das ações, fluxos alternativos e pós-condições. Dessa forma, depois de implementada a funcionalidade, é possível até mesmo realizar um teste contra seu caso de uso para ver se tudo está ok. A **Tabela 1** mostra a documentação para o caso de uso Manter Clientes.

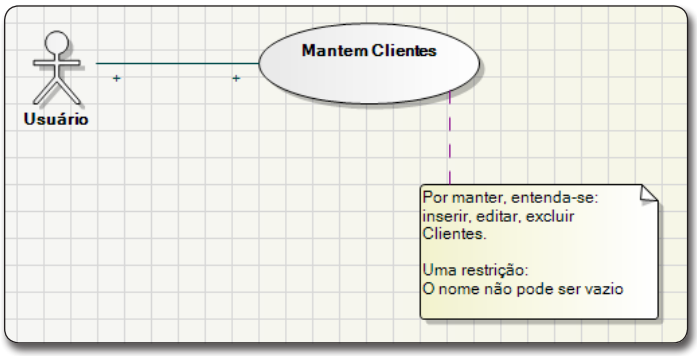

Figura 5. Caso de uso Manter Clientes

Caso de Uso	Manter Clientes
Descrição Geral	Realizar o cadastro dos clientes, com operações para incluir, alterar e excluir informações. Campos que serão gravados: Nome, Endereço, Cidade, Uf, Cep.
Ator principal	Usuário
Pré-condição	Sistema estar em execução
Fluxo Principal	1. Usuário clica em inserir 2. Usuário preenche os campos solicitados 3. Usuário confirma informações 4. Sistema valida os dados
Fluxo Alternativo	4.1. O sistema detecta que nome está em branco 4.2. Sistema alerta usuário e não confirma operação
Pós-condição	Cliente é salvo

Tabela 1. Documentação de Caso de Uso

Para efeito de exemplo, abordamos somente a situação de inclusão, mas seria preciso descrever os possíveis fluxos para as ações que poderia ser tomadas.

Descrevendo as entidades do projeto

A UML oferece um diagrama específico para isso, o diagrama de classes. Ele é uma representação da estrutura e relacionamentos que as classes que representam as entidades podem possuir. Esse diagrama se assemelha muito ao MER (Modelo de Entidade-Relacionamento). Uma classe entidade pode ser definida como uma classe que contém informações que precisam ser salvas. Para ficar mais claro, baseando-se nesse projeto mesmo, uma entidade seria uma classe TCliente que conteria as propriedades Nome, Endereço, Cidade, Uf e Cep.

Como o projeto não é totalmente orientado a objetos, ele não contém essa classe. Essa divergência é o principal motivo para

os desenvolvedores Delphi ignorarem a UML e seus diagramas no dia a dia. É muito importante ter maturidade o suficiente para entender que, todo processo deve ser adaptado a alguma realidade. Assim, é possível fazer uso da UML em seu processo de desenvolvimento e utilizar somente o que é útil. Neste caso o diagrama de classes não faz sentido, mas um MER (modelo entidade relacionamento) sim. Veja, podem-se mesclar conceitos, o importante é que tudo fique claro para o desenvolvedor.

Descrevendo a interatividade

Um diagrama de sequência é utilizado para demonstrar como deve ser feita a interação entre as classes da funcionalidade. Apesar de não termos classes de entidades, todo o restante do projeto reside em classes. Então podemos modelar como elas devem se comunicar durante as ações. Para compreender melhor, temos na **Figura 6** o diagrama de sequência baseado no caso de uso Manter Cliente. Ele foi incluído ao projeto clicando-se com o botão direito no projeto (na guia Model View) e escolhendo o item Add>Other Diagram>Sequence Diagram. Ao adicionar um novo diagrama do tipo Sequência, o IDE do Delphi também disponibiliza na Tool Palette os elementos possíveis.

Os diagramas de sequência são muito importantes porque esclarecem os papéis dos objetos dentro de um fluxo de operações, permitindo que se identifique de forma fácil as responsabilidades. O diagrama mostra como os objetos se comunicam através das mensagens (métodos). Um detalhe importante é que eu não preciso ter a tela pronta pra criar esse diagrama. Lembre-se, toda essa representação é utilizada para descrever um funcionamento, assim, podemos descrever tudo isso e encaminhar para um desenvolvedor implementar.

Ao analisar o diagrama da **Figura 6** vemos que temos o papel de um usuário representado nele. O usuário acessa os objetos do sistema. No caso, temos o formulário principal. Os objetos são representados por um nome que não necessariamente deve ser o mesmo que existirá ou existe no código fonte. As ações que ele acessa são representadas pelas setas, e são os métodos do objeto para onde a seta aponta. Essas ações são numeradas em uma possível ordem de execução. Pode-se observar a sequência gerada.

A ação número 1 representa a criação do formulário, assim, ela não tem uma descrição, mas poderia ter. Seguindo pela sequência, quando o formulário é criado, ele associa o evento OnChangeState a um método do formulário. Então, nas ações seguintes vemos a interação do usuário ao clicar nos botões. Por exemplo, o usuário dispara o evento btnInserir.OnClick, contudo, essa mensagem possui uma pré-condição que está representada entre "[]". No caso essa mensagem só é enviada se a condição "not Editando" for satisfeita. Se for satisfeita, é mostrado então o que é feito no evento, o formulário acessa o cdsCliente do Data Module e executa o método Append. Um desenvolvedor, que conheça UML, ao ler esse diagrama entenderá que precisará criar a condição e já entenderá como implementar essa troca de mensagens. Outra situação interessante está na sequência da ação 5. O item 5.1 informa a existência de um procedimento de validação, esse foi citado na descrição do caso de uso. Como foi dito, um diagrama de sequência se baseia em um caso de uso. Se a validação não é satisfeita, o fluxo da mensagem retorna ao usuário através de uma mensagem (item 5.2). De uma forma geral o comportamento fica muito claro. Mas se já temos a descrição do caso de uso, que também expõe a interação, porque não utilizar apenas ela?

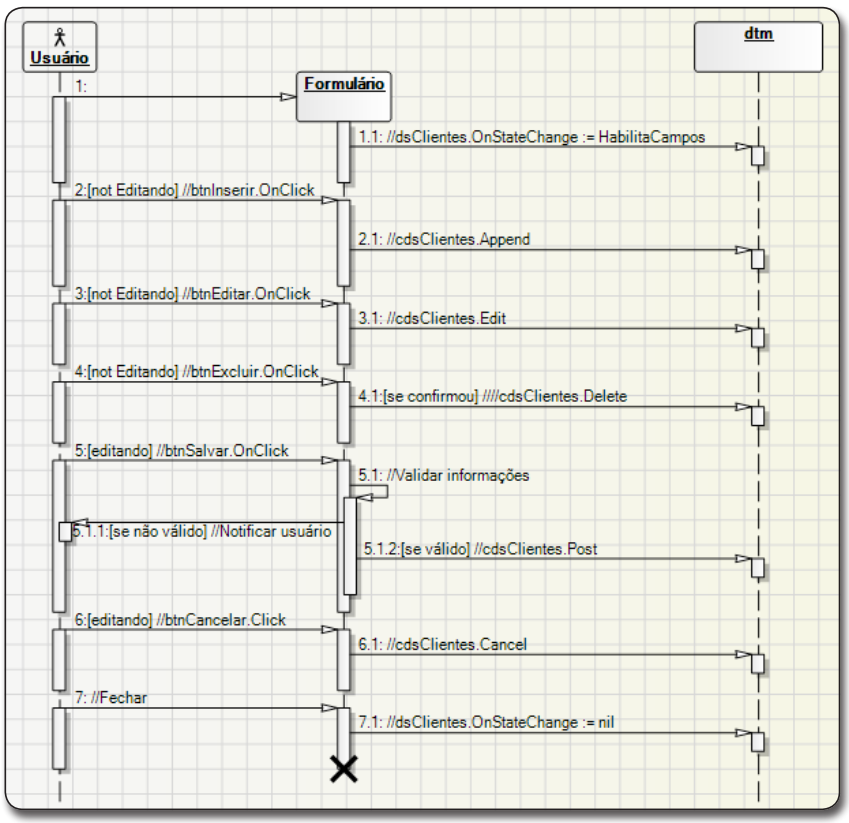

Figura 6. Diagrama de sequência do caso de uso Manter Clientes

A resposta vai depender de N fatores. Enquanto que um caso de uso e sua descrição mostram o ponto de vista do usuário, isto é, não expõe nada técnico, no diagrama de sequência isso é diferente. Nele temos condições de dizer que haverá um formulário, um Data Module, um objeto ClientDataset a ser utilizado, etc. Seu uso e detalhamento estão estritamente ligados à necessidade de documentação, e ao uso da mesma. Se for para ser utilizada como base de implementação para um desenvolvedor novato, o detalhamento de tipos e nomes de objetos, métodos, se faz necessário. Mas se for apenas pra discutir, mostrar um funcionamento complexo para outros desenvolvedores, esse detalhamento já não é requerido.

Modelando estado

Temos definido na **Figura 6** uma pré-condição chamada Editando, ela claramente é um estado geral da funcionalidade e impacta nos objetos envolvidos. O diagrama de máquina de estado pode ser utilizado para descrever o que acontece quando esse estado é alterado. Um diagrama de máquina de estado é representado pelos seguintes elementos: Estados, Transições, Condições de Guarda e Ações.

Um estado é a condição de um objeto em que ele realiza alguma atividade ou espera de evento. Um estado possui nome, ações de entrada e saída, pode possuir ainda um sub estado. Uma transição é a ligação entre dois estados, indicando que um objeto no primeiro estado executará certas ações e entrará no segundo estado. A transição possui um estado de origem, um gatilho (trigger) de evento, esse torna a transição passível de ser acionada. Possui também a condição de guarda, que é uma expressão booleana que é avaliada quando a transação é disparada. É utilizada para identificar se a transação será disparada ou não. Temos também ação e o estado de destino da transação. Os elementos condições de guarda e ações são similares aos do elemento transação.

Para adicionar esse diagrama ao projeto, clica-se com o botão direito no projeto (na guia Model View) e escolhe-se o item Add>Other Diagram>StateChart Diagram. Novamente o IDE do Delphi disponibiliza na Tool Palette os elementos possíveis. A **Figura 7** mostra o diagrama máquina de estado para o caso de uso Manter Clientes.

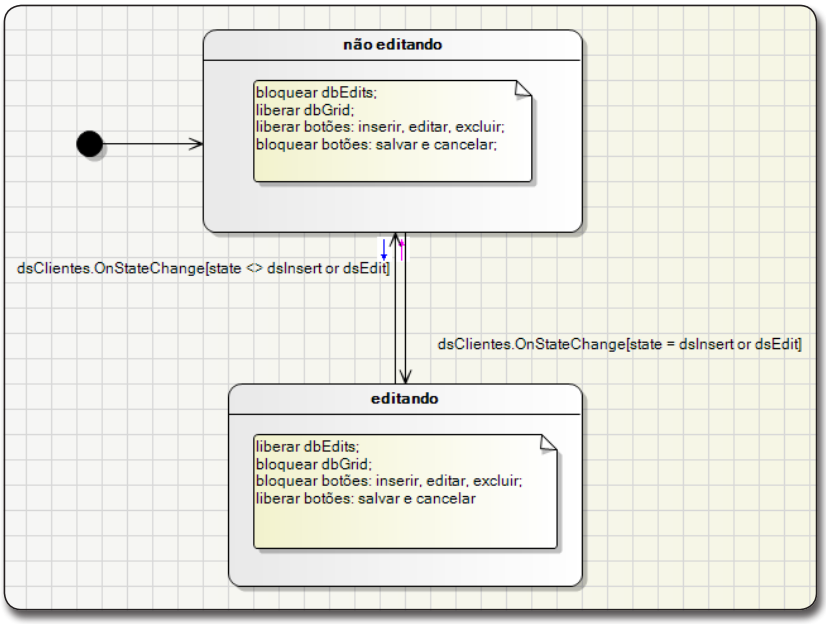

Figura 7. Máquina de Estados

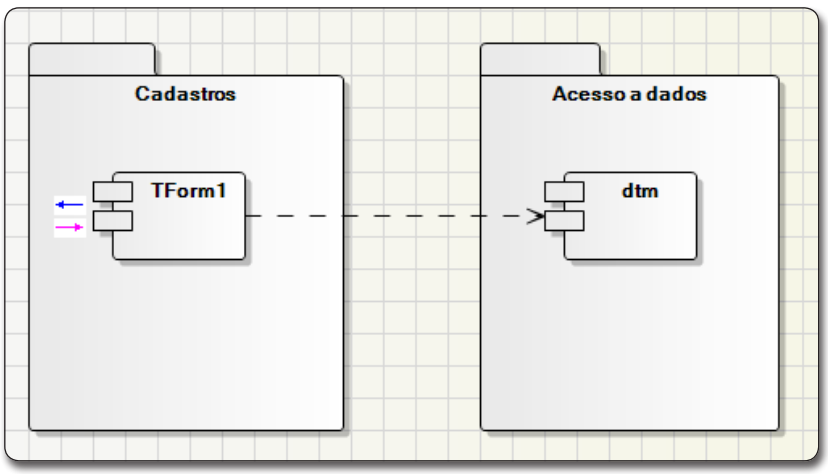

Figura 8. Diagrama de componentes

O círculo negro indica o ponto inicial da aplicação, o estado inicial. Assim, sabe-se que o formulário inicia no estado não editando. Os estados são representados pelos retângulos, que incluem seu nome e as ações, que nada mais são que um conteúdo textual. A mudança dos estados é representada pelas setas direcionais que no caso, estão indicando que o estado pode ir de não editando para editando e vice-versa. O detalhe é que nessas setas é indicado o evento causador da mudança do estado e sua condição guarda. O evento é o OnStateChange e a condição é que o estado seja de inserção ou edição para que a transição para editando aconteça e para retornar ao estado não editando, que o estado seja diferente de inserção e edição.

Organizando em componentes

Quando um sistema é muito grande e se faz necessário ter uma visualização organizada das classes em relação às suas dependências ou a sua função dentro da regra de negócios, podemos utilizar o diagrama de Componentes. A **Figura 8** mostra o diagrama de componentes do projeto de exemplo. Ele foi inserido no projeto pelo menu Add>Other Diagram>Component Diagram da guia Model View.

Nele as classes foram agrupadas em subsistemas, Cadastros e Acesso a dados. Em um sistema real, de maior complexidade, podemos encontrar subsistemas como Pagamentos, Emissão de E-mails, Controle de Usuários, etc. Para dentro de cada um deles é arrastado um componente que os compõem. Temos o TForm1 para Cadastros e dtm para Acesso a dados. De forma semelhante nesse momento, em um sistema real, estariam detalhados os formulários, units, classes, de acordo com sua categoria de negócio. Também podemos ver o que depende do que. Essa dependência é expressa pela linha tracejada, que na **Figura 8,** diz que Cadastros depende de Acesso a dados, mas de uma forma mais granular, também diz que TForm1 depende de dtm. Essa é uma ótima forma para mostrar para um novo integrante de um time como está organizado o projeto com o qual irá trabalhar.

Conclusão

Neste artigo vimos que é possível sim utilizar a UML mesmo para sistemas que não são orientados a objetos em sua totalidade. Seus diagramas permitem uma imersão imediata a um determinado ponto de um sistema, fazendo com que haja uma integração de compartilhamento de conhecimento. Apesar de não conseguirmos utilizar o diagrama de classes, que é o mais falado quando o assunto é UML, o diagrama de sequência se mostrou útil para entender completamente como uma funcionalidade se comporta, juntamente com o caso e descrição de uso. A máquina de estados então deixou claro como e o que deve ser feito quando um estado muda, e o de componentes exibiu uma visão geral do sistema.

Ao analisar os procedimentos de muitas empresas, que criam um manual textual para ser apresentado para novos desenvolvedores de uma equipe, e comparar com esses poucos diagramas apresentados aqui, o que surtirá um efeito mais rápido para essa pessoa que nunca viu o projeto? Fica essa pergunta a você leitor.

Paulo Quicoli
pauloquicoli@gmail.com
Editor da revista ClubeDelphi e editor técnico .Net Magazine. Formado em processamento de dados pela FATEC-TQ. Atua como analista programador de sistemas da Siplan Control-M (www.siplancontrolm.com.br) unidade Jaboticabal, e professor da FATEC, campus Taquaritinga. Blog: http://quicoli.worpress.com

Dê seu feedback sobre esta edição!

A ClubeDelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/clubedelphi/feedback

FÓRUM
DEVMEDIA
O lugar perfeito para você ficar por dentro de tudo o que acontece nas tecnologias do mercado atual

No fórum da DevMedia você irá encontrar uma equipe disponível e altamente qualificada com consultores e colaboradores prontos para te ajudar a qualquer hora e sobre qualquer assunto. Temos as salas de Java, .NET, Delphi, Banco de Dados, Engenharia de Software, PHP, Java Script, Web Design, Automação comercial, Ruby on Rails e muito mais!

ACESSE AGORA
www.devmedia.com.br/forum

DevMedia

Somos tão apaixonados por tecnologia que o nome da empresa

diz tudo.

Porta 80 é o melhor que a Internet pode oferecer para sua empresa.

Já completamos 8 anos e estamos a caminho dos 80, junto com nossos clientes.

Adoramos tecnologia.
Somos uma equipe composta de gente que entende e gosta do que faz,

assim como você.

Estrutura

100% NACIONAL. Servidores de primeira linha, links de alta capacidade.

Suporte diferenciado

Treinamos nossa equipe para fazer mais e melhor. Muito além do esperado.

Serviços

Oferecemos a tecnologia mais moderna, serviços diferenciados e antenados com as suas necessidades.

1-to-1

Conhecemos nossos clientes. Atendemos cada necessidade de forma única.
Conheça!

Porta 80
WEB HOSTING

Hospedagem | Cloud Computing | Dedicados | VoIP | Ecommerce | Aplicações | Streaming | Email corporativo

porta80.com.br | comercial@porta80.com.br | twitter.com/porta80

SP 4063-8616 | RJ 4063-5092 | MG 4063-8120 | DF 4063-7486

TENHA ACESSO ILIMITADO A TODO CONTEÚDO DO SITE DEVMEDIA

SEJA UM ASSINANTE MVP!

Tenha acesso garantido a qualquer tema que você possa necessitar!

VIDEOS -AULAS

O assinante MVP acessa todas as vídeo-aulas do site DevMedia. Nenhum outro produto dá essa possibilidade. São mais de **7.000 video-aulas.**

CURSOS

Nosso site possui atualmente **+ de 250 cursos online** com a qualidade que você já conhece e confia. Com o MVP você terá acesso a todos eles.

REVISTAS

Assinante MVP acessa todas as nossas revistas na versão digital, incluindo todas as edições anteriores. Quando pintar aquela dúvida você terá mais de **3.500 artigos para consultar!**

ASSINATURA MVP MOST VALUABLE PROFESSIONAL

A assinatura MVP te dá a oportunidade de aumentar seu conhecimento em diversos assuntos, com qualidade e economia.

Invista na sua carreira se torne um profissional ainda mais completo!

POR APENAS R$59,90 POR MÊS

DevMedia

Acesse: www.devmedia.com.br/mvp